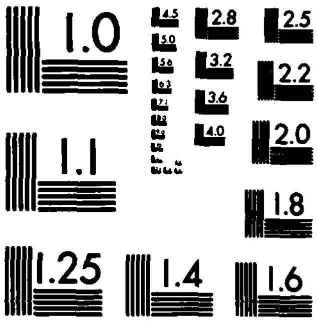
AD-A145 558 PARAMETRIC ANALYSIS FOR GENERALIZED NETWORK FLOW PROBLEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH M E BAUM MAY 84 AFIT/CI/NR-84-38D 1/3

UNCLASSIFIED F/G 12/1 NL


1.0
4.5
5.0
5.6
6.3
7.1
8.0
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

AD-A145 558

DTIC FILE COPY

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 84-38D	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Parametric Analysis for Generalized Network Flow Problems		5. TYPE OF REPORT & PERIOD COVERED ~~THESIS~~/DISSERTATION
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Michael Ernest Baum		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: The University of Texas at Austin		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS AFIT/NR WPAFB OH 45433		12. REPORT DATE May 1984
		13. NUMBER OF PAGES 187
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DTIC
ELECTE
SEP 17 1984
S D
B

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

APPROVED FOR PUBLIC RELEASE: IAW AFR 190-17

[illegible]
LYNN E. WOLAVER
Dean for Research and
Professional Development
AFIT, Wright-Patterson AFB OH

24 July 84

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

ATTACHED

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASS

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

84 09 13 002

PARAMETRIC ANALYSIS FOR GENERALIZED NETWORK FLOW PROBLEMS

Publication No. ______

Michael Ernest Baum, Ph.D.
The University of Texas at Austin, 1984

Supervising Professor: Paul A. Jensen

A generalized network flow problem can be efficiently solved with state of the art computer codes. However, because of the uncertainty or inaccuracy of initial problem input data, the solution found may need reexamination.

Parametric analysis allows the management scientist to vary any of the right-hand side restrictions, either external flow or capacities, and determine the series of bases so the solution remains optimal. This study uses the special structural properties of the generalized network flow problem to iteratively change the network flows, until no further changes in the basis can be made.

Additionally, two dual-incremental flow algorithms based on parametric analysis are developed. A series of test problems were randomly generated and computational results were compared to two primal generalized network computer codes and a published dual incremental computer code.

[illegible]

AFIT/CI/NR 84-38D

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to ascertain the value and/or contribution of research accomplished by students or faculty of the Air Force Institute of Technology (AU). It would be greatly appreciated if you would complete the following questionnaire and return it to:

AFIT/NR
Wright-Patterson AFB OH 45433

RESEARCH TITLE: Parametric Analysis for Generalized Network Flow Problems

AUTHOR: Michael Ernest Baum

RESEARCH ASSESSMENT QUESTIONS:

1. Did this research contribute to a current Air Force project?

( ) a. YES ( ) b. NO

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not?

( ) a. YES ( ) b. NO

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency achieved/received by virtue of AFIT performing the research. Can you estimate what this research would have cost if it had been accomplished under contract or if it had been done in-house in terms of manpower and/or dollars?

( ) a. MAN-YEARS ______ ( ) b. $______

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3. above), what is your estimate of its significance?

( ) a. HIGHLY SIGNIFICANT ( ) b. SIGNIFICANT ( ) c. SLIGHTLY SIGNIFICANT ( ) d. OF NO SIGNIFICANCE

5. AFIT welcomes any further comments you may have on the above questions, or any additional details concerning the current application, future potential, or other value of this research. Please use the bottom part of this questionnaire for your statement(s).

NAME GRADE POSITION

ORGANIZATION LOCATION

STATEMENT(s):

38

PARAMETRIC ANALYSIS FOR GENERALIZED NETWORK FLOW PROBLEMS

APPROVED BY SUPERVISORY COMMITTEE:

[illegible]

[illegible]

[illegible]

[illegible]

84 09 13 002

To my Mother who I wish could
have seen this work completed.

To my Father on his 75th birthday.

To my family and extended family for
all their encouragement and understanding.

To my wife, whose steadfast support kept
me going when there seemed to be no light
at the end of the tunnel.

PARAMETRIC ANALYIS FOR GENERALIZED NETWORK FLOW PROBLEMS

by

Michael Ernest Baum, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

MAY 1984

ACKNOWLEDGEMENTS

The author would like to express his sincere gratitude to Dr. Paul A. Jensen, Chairperson of his Doctoral Committee, for his invaluable guidance, advice, assistance, support, and patience during the past year and one-half. The author would also like to acknowledge the other members of his Doctoral Committee, Dr. J. Wesley Barnes, Dr. Jay Brennan, Dr. Leon S. Lasdon, and Dr. William G. Lesso for their constructive comments, suggestions, and words of encouragement.

Finally, a special thanks is due to my wife Sigrid. Her constant encouragement and patience were an immeasurable contribution to the success of this work.

38

PARAMETRIC ANALYSIS FOR GENERALIZED NETWORK FLOW PROBLEMS

Publication No. ______

Michael Ernest Baum, Ph.D.
The University of Texas at Austin, 1984

Supervising Professor: Paul A. Jensen

A generalized network flow problem can be efficiently solved with state of the art computer codes. However, because of the uncertainty or inaccuracy of initial problem input data, the solution found may need reexamination.

Parametric analysis allows the management scientist to vary any of the right-hand side restrictions, either external flow or capacities, and determine the series of bases so the solution remains optimal. This study uses the special structural properties of the generalized network flow problem to iteratively change the network flows, until no further changes in the basis can be made.

Additionally, two dual-incremental flow algorithms based on parametric analysis are developed. A series of test problems were randomly generated and computational results were compared to two primal generalized network computer codes and a published dual incremental computer code.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 The Generalized Minimum Cost Network Flow Problem

In recent years there has been an increasing awareness that network flow programming is a powerful modeling and problem solving technique that can be applied to a wide range of physical and conceptual situations. The network flow programming model encompasses a variety of problem classes, including the shortest path problem, the maximum flow problem, the pure minimal cost flow problem, the convex network flow problem, and the generalized minimum cost flow problem. Among these classes, the generalized network is the problem that is considered in this research. The generalized transportation problem and generalized assignment problem are special cases of a generalized minimum cost flow problem.

The network of a generalized minimum cost flow problem is a directed graph defined by a set of arcs, M, with ordered pairs of nodes (i,j) as elements indexed by k. For each arc in the generalized minimum cost flow problem there is: a cost h_k for each unit of flow which passes through the arc; a lower bound $\underline{c}_k$, which is the minimal allowable flow that an arc can carry ; a capacity c_k, which is the maximum amount of flow that an arc can carry ; and a gain factor a_k which determines if flow is to be lost, generated, or conserved on the arc. Each node in the minimum cost flow problem is either a supply node where flow enters the network, a demand node where flow leaves the network, or a transshipment node where no flow enters or leaves the network. The required quantities of flow entering or leaving the network at the nodes are called external flows. A positive external flow enters the network at a node, and a negative external flow leaves the network at that node.

The goal of a generalized minimum cost flow problem is to determine how a commodity should be delivered through the arcs of a network, that is to choose the arc flows f_k, so that shipment cost will be minimized and two kinds of constraints are satisfied. The two constraints to be satisfied are : 1) that flow be conserved at each node, which is to say that flow leaving a node from the arcs of a network minus flow entering the node on the arcs of the network must equal the external flow at the node; and 2) that the flow on each arc be between its lower bound and capacity. The algebraic representation of the model can be expressed as a bounded linear programming problem with a constraint for each node and a variable for each arc.

The algebraic model is stated as follows:

$$\text{Min} \quad \sum_{k=1}^{m} h_k f_k$$

$$\text{S.T.} \quad \sum_{k \in M_{Oi}} f_k - \sum_{k \in M_{Ti}} a_k f_k = b_i \qquad i=1,\ldots,n-1$$

$$\underline{c}_k \leq f_k \leq c_k \qquad k \in M$$

or in matrix form

$$\text{Min} \quad HF$$

$$\text{S.T.} \quad AF = b$$

$$\underline{C} \leq F \leq C$$

where

f_k is the amount of flow on arc k

h_k is the shipping cost for each unit of flow on arc k

$\underline{c}_k$ is the minimum amount of flow on arc k

c_k is the maximum amount of flow on arc k

a_k is the gain parameter on arc k

b_i is the external flow at node i (positive for incoming flows, negative for outgoing flows)

M_{Oi} is the set of arcs which originate at node i

M_{Ti} is the set of arcs which terminate at node i

F,H,$\underline{C}$,C, and b are respectively the collections of f_k, h_k, $\underline{c}_k$, c_k, and b_k. A is the node arc incidence matrix of the generalized minimum cost flow model. An example of the A matrix will be shown in Chapter 4.

1.2 Parametric Programming

An important part in the analysis of any large mathematical programming problem is the study of how the solution changes with variations of the problem data. There may be questions concerning the accuracy of the data or a simple desire to see how the solution changes as elements of the solution change. This problem gives rise to the area of sensitivity analysis. When performing sensitivity analysis, one determines the region over which the restrictions may by changed so as to maintain an optimal solution. This analysis may be extended to parametric programming, where a series of optimal solutions are generated as the conditions are continuously changed. This allows the analyst to be able to answer a series of "what if" questions about the problem. The flexibility created by allowing changes in any part of the problem data leads to a much more powerful analytic technique.

In particular the analyst may wish to vary some of the right-hand side restrictions. These might include variations of the external flows (b) or the capacity restrictions (C). We can state the parametric programming problem for varing the external flows as follows:

$$\begin{array}{ll} \text{Min} & HF \\ \text{S.T.} & AF = b + \Theta \hat{b} \\ & \underline{C} \leq F \leq C \\ & 0 < \Theta \leq \Theta_{\max} \end{array}$$

where Θ is the change to be accomplished in the external flow vector b. Similarly we can state the parametric programming problem for the variation of the capacity vector as

$$\begin{array}{ll} \text{Min} & HF \\ \text{S.T.} & AF = b \\ & \underline{C} \leq F \leq C + \lambda \hat{C} \\ & 0 < \lambda \leq \lambda_{\max} \end{array}$$

where λ is the change to be accomplished in the capacity vector C. We can also give a concise verbal description of the parametric programming problem as follows:

Given: An optimal solution to a generalized minimum cost network flow problem and a set of parameters of the problem which are to be varied.

Problem: Determine a series of optimal solutions to the original problem as the parameters are continuously varied.

By Choice of: How much flow to send along each arc of the network.

Subject to: External flow restrictions must be met; conservation of flow at each node must be maintained; lower and upper bounds on each arc must not be violated.

The significance of this solution technique is that it will allow the analyst to examine a range of optimal solutions for the problem being modeled. Secondly it opens the avenues to the solution of problems which have imbedded networks as subproblems in which either capacities or external flows are being varied. The solution algorithm could take the previous iterations' optimal solution and iterate by use of the parametric sensitivity algorithm which will be presented.

1.3 Study Outline

In this research, an attempt is made to develop a parametric programming algorithm which applies to the generalized network flow problem. A detailed review of past literature relative to the history of linear programming, generalized network programming, and the solution technique is presented in Chapter 2. In Chapter 3 a summary of linear programming theory relevant to this problem is discussed. Chapter 4 is a review of pertinent network flow terminology and theory. Chapter 5 presents a detailed description and explanation of the solution technique as it is applied to the generalized network flow problem. Finally, in Chapter 6, two new dual-incremental flow algorithms are developed. Computational results comparing these codes to two primal codes and a published dual-incremental code are presented.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The generalized minimum cost network flow problem is an important class of network flow problems in which efficient solution algorithms exist. The generalized network flow problem is a special case of a bounded variable linear program. Since linear programming has played such an important role in the development of network flow programming, we first review the history of linear programming before further discussion of generalized network flow literature.

2.2 Linear Programming Background

Some of the original works of linear programming theory can be traced back to 1823 when Fourier experimented with homogeneous linear systems. These methods were similar to the later discovered simplex algorithm (28). During this same period the works of Gauss and Jordan also established some basic techniques for solving linear systems of equations. P. Gordan appears to be the first to recognize that a linear system of homogeneous equations has an associated dual homogeneous problem and that non-negative variables in these systems of equations possesses a solution with at least one variable positive if the dual possesses no solution with strict inequalities. Steimke expanded this result to include all positive variables (21). Applying Fourier's principles, Minkowski concluded that in a homogeneous system, the general system can be formed as a non-negative linear combination of finite extreme point solutions. Ville, Von Neuman, and Morgenstein presented relationships between primal and dual solutions in the late 1930's and early 1940's, but it was Kantorovich, in 1939, who came extremely close to developing the first linear programming algorithm. He outlined solution methods to a class of problems consisting of simultaneous outputs (49). His initial approach was based on an initial feasible dual solution.

The most influential work appeared to be that of Neyman and Pearson (63), in 1936, whose development of some duality and linear inequality theories led to the significant discovery of Dantzig (21). The first primal algorithmic procedure for linear programming problems was the simplex method developed by Dantzig in 1947. Shortly, thereafter, Von Neuman expressed the additional importance of duality in solving linear systems (78). In 1951 Tucker, Kuhn, and Gale developed a duality theorem. Also in 1951, Dantzig and Orden proved that the optimal solution for the primal provided a corresponding optimal for the dual. Lemke, 1954, discovered the dual simplex algorithm. He also added an important note on the complementary slackness conditions between the primal and dual solutions (56). Charnes, et al. (18,19) pioneered the application of linear programming techniques to the industrial world, besides publishing numerous articles which contributed to linear programming theory. Continued search for new algorithmic approaches resulted in Orchard-Hay's composite simplex which is used in those cases where a solutiôn was found to be neither primal or dual feasible (64).

These results formed the basis of linear programming. During the 1950-1970's with the rise of improved computer technology, much of the research done was to improve the computational efficiency of the simplex on large scale programming applications. The introduction of the revised-simplex technique of Dantzig, Orchard-Hays and others led to more efficient numerical methods. Dantzig and Van Slyke (22) improved the computational efficiency of a special class of linear programs with generalized upper bounds (GUB) constraints. Hellerman and Rarick (41) and Forrest and Tomlin (27) worked on numerical stabilty for reinverting the basis. Schrage (68,69) and Todd (75) developed special algorithms for applications of the simplex method to linear programs with variable upper bounds (VUB).

The refinement of data storage and retrieval, as well as more efficient computational methods, have been a springboard in recent years for the methods mentioned above to be applied to a wide field of complex large scale linear programs.

2.3 Generalized Networks

The first computational procedures for the generalized network problem related to the generalized transportation problem. This application has a bipartite structure and uncapacitated arcs. Dantzig (21) and Charnes and Cooper (17) described early attempts at adapting the simplex method to this special problem. Eisemann (24), Lourie (57), and Balas and Ivanescu (11) implemented the use of primal procedures in relation to the stepping stone method of the pure transportation problem. Eisemann (24) additionally discussed the use of a dual solution approach to the problem. All these methods use the matrix representation of the transportation problem and are based on the bipartite nature of the graph.

Jewell (47) introduced a procedure similar in concept to the out of kilter algorithm (26) to solve the generalized minimum cost flow problem on a general network with capacities. Minieka (61) modified Jewell's algorithm to guarantee finite termination.

Johnson (48) suggested the use of the three pointer representation of the basis in the generalized network problem. Maurras (59) and Glover, Klingman, and Stutz (37) utilized a three pointer representation to implement primal algorithms for the generalized minimum cost flow problem. Additional computational simplifications were made to the generalized transportation problem by Glover and Klingman (36). Jensen and Bhaumik (44) described a dual incremental approach.

Hultz and Klingman (43) provided an improved dual feasible start and made computational comparisons on pure network test problems. Glover, et al., (38) presented an application paper on the use of generalized networks and computational testing with their generalized network code NETG. Elam, Glover, and Klingman (29) presented a primal simplex algorithm which is specially designed to reduce the number of degenerate pivots. Jensen and Barnes (46) developed a generalized primal code, called PGAINS, and a dual incremental code call INCREMG. Adolphson (2) extented the preorder thread to generalized models in his computer code AGENNET. Gibby, et al., (32) explored different pivoting strategies to improve the efficiency of primal codes.

The interest in the area of applications of generalized networks continues to expand, as does the size of the problems which are being solved by these solution methods. We have merely surveyed the major accomplishments in the area of generalized networks.

2.4 Sensitivity Analysis and Parametric Programming

During the history of linear programming many economists and specialists have met with failure when they have attempted to introduce linear programming into their operations. This has been based frequently on one of the following factors:

1. The uncertainty and inaccuracy of the initial data.
2. The problems of evaluation and interpretation as well as application and exploitation of the results in practice.

The problems mentioned in 1. gave rise to the area of sensitivity analysis in which one may test in what regions the right-hand side restrictions may be changed to maintain the optimality of the solution. The earliest use of parametric programming as applied to linear programs dates back to Gass and Saaty (30,31). Parametric programming enables us to compute all existing optimal basic solutions in relation to their dependence on the values of the components of the right-hand side. Gal (29) lists over 500 references in his bibliography in relation to either parametric or sensitivity analysis of linear programs.

The same problem of uncertainty and inaccuracy of the initial data should lead to similar problems in network solution analysis by today's managers. However these problems have not led to a great deal of research interest in this area. Thus, while there exist a large number of references in the linear programming area, there are relatively few references in the area of parametric analysis for network flow problems. It appears that the initial research in a related are was done by Srinivasan and Thompson (72,73) when they applied operator theory to parametric programming for the transportation problem. In a series of articles Balachandran and Thompson (5,6,7,8) apply operator theory to the generalized transportation problem. Srinivasan and Thompson (74) develop a theory of cost operator algorithms for transportion problems. In these methods parametric programming is use to solve the transportation problem to optimality. Balachandran (10) developes an operator theoretic approach for a generalized transportation problem with stochastic demands. Balachandran, Srinivasan and Thompson (9) demonstrate the application of their operator theory of parametric programming for both transportation and generalized transportation problems. Ahuja (3) developes a ranging analysis for pure capacitated transshipment problems. Adolphson (1), in a private communication to the author, describes a procedure for ranging analysis of the right-hand side, cost, and gains factors for a generalized network flow problem which has a computation time of order m (where m is the number of arcs).

CHAPTER 3

BACKGROUND IN LINEAR PROGRAMMING

3.1 Introduction

Since a capacitated generalized network is a special case of a linear program, it seems appropriate to discuss the fundamental ideas behind the linear programming simplex method. In this chapter we will first review the bounded variable simplex method, give a brief discussion of the dual-simplex method, and lastly develop the background for perturbation of the right-hand side of the constraints.

3.2 Statement of the Problem

Using matrix notation, the linear programming problem with simple upper bounds may be stated in the form:

$$\text{Min } Z = H^T F \tag{3.1}$$

$$\text{S.T.} \quad AF = b \tag{3.2}$$

$$0 \leq F \leq C. \tag{3.3}$$

While the above notation is not standard for linear programming, it follows the network notation used by Ford and Fulkerson (26). F is the vector of decision variables. H is the vector of variable costs. The right-hand side values are represented by the vector b, and C is the vector of capacities or upper bounds. There are n variables and m equations.

Equation (3.1) is called the objective function and equations (3.2) and (3.3) are called constraints. A is an m by n matrix with $n > m$. We will assume the equations are linearly independent, that is, we do not have any equation which is a multiple of one or more of the other equations. Inequality constraints may be made into equality constaints as in equation (3.2) by use of "slack variables" which are added to the left-hand side and whose values represent the difference between the original left-hand side and the right-hand side. They are called "logical variables"; the original variables are called "structural variables" as they have more physical significance.

The main concept behind upper bounding is that any nonbasic variable f_R may either be at zero of its upper bound c_R. This eliminates the need to represent the upper bound on a variable as a separate constraint, thus saving a row in the constraint matrix for each upper bound. This reduces m, the size of the basis matrix, with the resultant savings in both running time and storage requirements. The price paid for such savings is that the steps of the simplex algorithm become slightly more complicated.

We can use equation (3.2) to solve for m variables called the basic variables in terms of the other (n-m) variables called the nonbasic variables. We partition A into the submatrices B, R_{LB}, R_{UB}

$$A = [\ B \mid R_{LB} \mid R_{UB}\ ] \tag{3.4}$$

where B is an m by m nonsingular submatrix formed from the columns of the basic variables called the basis matrix; R_{LB} and R_{UB} constitute the columns of the nonbasic variables at zero and the columns of the nonbasic variables at their upper bounds, respectively.

We can express equation (3.2) in the form

$$BF_B + R_{LB}F_{LB} + R_{UB}F_{UB} = b \tag{3.5}$$

where F has been partioned into

$$F = \begin{bmatrix} F_B \\ \hline F_{LB} \\ \hline F_{UB} \end{bmatrix} \tag{3.6}$$

corresponding to the partitioning of A.

Equation (3.5) can be rearranged to the form

$$BF_B = b - R_{LB}F_{LB} - R_{UB}F_{UB} \tag{3.7}$$

and since B is nonsingular

$$F_B = B^{-1}b - B^{-1}R_{LB}F_{LB} - B^{-1}R_{UB}F_{UB}. \tag{3.8}$$

We have thus solved for m of the variables (F_B) in terms of the other (n-m) variables (F_{LB}, F_{UB}), where

F_B	is called the vector of basic variables
F_{LB}	is called the vector of nonbasic variables at zero
F_{UB}	is called the vector of nonbasic variables at the upper bound.

A basic solution is defined as one in which the nonbasic variables are set at either their upper bound, C_R, or zero. A basic feasible solution is one in which all the terms of the vector F_B are nonnegative.

3.3 The Transformed Problem

As we partitioned F into (F_B, F_{LB}, F_{UB}), we can also partition H into (H_B, H_{LB}, H_{UB}). We can now rewrite equation (3.1) as

$$\text{Min } Z = H_B{}^T F_B + H_{LB}{}^T F_{LB} + H_{UB}{}^T F_{UB}. \tag{3.9}$$

From equation (3.8) we have an expression for F_B in terms of F_{LB} and F_{UB} and can write the objective function as

$$\begin{aligned} Z = {} & H_B{}^T (B^{-1}b - B^{-1}R_{LB}F_{LB} - B^{-1}R_{UB}F_{UB}) \\ & + H_{LB}{}^T F_{LB} + H_{UB}{}^T F_{UB}. \end{aligned} \tag{3.10}$$

Rearranging terms

$$\begin{aligned} Z = {} & H_B{}^T B^{-1} b - (H_B{}^T B^{-1} R_{LB} - H_{LB}{}^T) F_{LB} \\ & - (H_B{}^T B^{-1} R_{UB} - H_{UB}{}^T) F_{UB}. \end{aligned} \tag{3.11}$$

Thus we can express our original problem, equations (3.1 - 3.3), in the form

$$\text{Min } Z = H_B{}^T B^{-1} b - (H_B{}^T B^{-1} R_{LB} - H_{LB}{}^T) F_{LB}$$
$$- (H_B{}^T B^{-1} R_{UB} - H_{UB}{}^T) F_{UB} \quad (3.12)$$
$$\text{S.T.} \quad F_B = B^{-1} b - B^{-1} R_{LB} F_{LB} - B^{-1} R_{UB} F_{UB} \quad (3.13)$$
$$F_B \geq 0.$$

This is called the transformed problem, or sometimes referred to as the current tableau. If we let

$$\beta = B^{-1} b$$

and

$$\pi^T = H_B B^{-1}$$

the transformed problem can be expressed as

$$\text{Min } Z = H_B{}^T \beta - (\pi R_{LB} - H_{LB}{}^T) F_{LB}$$
$$- (\pi R_{UB} - H_{UB}{}^T) F_{UB} \quad (3.14)$$
$$\text{S.T.} \quad F_B = \beta - B^{-1} R_{LB} F_{LB} - B^{-1} R_{UB} F_{UB} \quad (3.15)$$
$$F_B \geq 0.$$

Since $F_{LB} = 0$, F_B takes on the numerical value of $\beta - B^{-1} R_{UB} F_{UB}$ and the objective function takes on the value of $H_B \beta - (\pi R_{UB} - H_{UB}) F_{UB}$. We leave the right-hand terms of the equations in the transformed problem even though $F_{LB} = 0$, since they indicate what happens to Z and F_B as one of the elements of F_{LB} is increased from zero or one of the elements of F_{UB} is decreased from its upper bound.

3.4 Conditions for Optimality

The row vectors

$$(\pi^T R_{LB} - H_{LB}^T) \tag{3.16}$$

$$(\pi^T R_{UB} - H_{UB}^T) \tag{3.17}$$

which premultiply F_{LB} and F_{UB} in equation (3.14), the objective row of the transformed problem, are called the vectors of reduced costs, since they indicate how much the objective Z decreases as F_{LB} increases from zero or the amount Z increases as F_{UB} decreases from C_{UB}.

Let us denote the jth element of the reduced cost vectors, equations (3.16-3.17) by

$d_{jLB} = (z_{jLB} - h_{jLB})$ for nonbasic variables at zero

$d_{jUB} = (z_{jUB} - h_{jUB})$ for nonbasic variables at upper bounds where h_{jLB} is the jth element of H_{LB} and h_{jUB} is the jth element of H_{UB}, and

$$z_{jLB} = (\pi^T R_{LB})_j = \pi^T a_{jLB} \tag{3.18}$$

$$z_{jUB} = (\pi^T R_{UB})_j = \pi^T a_{jUB} \tag{3.19}$$

where a_{jLB} and a_{jUB} are the jth columns of R_{LB} and R_{UB}, the nonbasic portion of A.

We see that if for f_{jLB} the corresponding reduced cost is positive, Z will decrease upon increasing f_{jLB} above zero, while if for f_{jUB} the corresponding reduced cost is negative, Z will decrease on decreasing f_{jUB} below c_j.

Thus if our transformed problem represents a basic feasible solution with $F_B = \beta - B^{-1}R_{UB}F_{UB} \geq 0$, $F_{LB} = 0$, and $F_{UB} = C_{UB}$, then the further condition for optimality is that all elements of the reduced cost vectors be negative (or zero) for F_{LB} and positive (or zero) for F_{UB}.

Summarizing these conditions:

Feasibility	$\beta_i - B^{-1}R_{iUB}F_{iUB} \geq 0$	$i=1,\ldots,m$
Optimality	$d_j \leq 0 \quad j \in LB$	
	$d_j \geq 0 \quad j \in UB$	

where

$$LB = \{j \in R \mid f_j \text{ is at } 0\}$$

$$UB = \{j \in R \mid f_j \text{ is at } c_j\} .$$

3.5 Pivoting

We consider the possibility of increasing only one nonbasic variable from zero or decreasing from its upper bound, while all others stay at zero or their upper bounds. If more than one reduced cost is positive for a nonbasic variable at zero or negative for a nonbasic variable at its upper bound, we have an arbitrary choice. It is customary in most introductory texts to consider the nonbasic variable with the "largest" reduced cost since this will decrease the value of the objective function the most per unit change. There are more elaborate ways to pick the entering variable, but we will not discuss them here (see reference 62).

Writing out the transformed problem column by column, it takes the form

$$\begin{aligned} \text{Min } Z &= H_B^T\beta - d_1(F_R)_1 - \dots - d_q(F_R)_q - \dots \\ &\quad - d_{n-m}(F_R)_{n-m} \qquad (3.20) \\ \text{S.T. } F_B &= \beta - \alpha_1(F_R)_1 - \dots - \alpha_q(F_R)_q - \dots \end{aligned}$$

$$- \alpha_{n-m}(F_R)_{n-m} \tag{3.21}$$

where

$$\alpha_j = B^{-1}a_j \qquad j = 1,\ldots,n-m. \tag{3.22}$$

Here we have chosen a nonbasic variable $(F_R)_q$ to increase from zero or to decrease from its upper bound. Since some of the nonbasic variables may be at their upper bound, β does not represent the current value of F_B. Rather, we have

$$F_B = \beta - \sum_{j\varepsilon UB} \alpha_j(F_R)_j \tag{3.23}$$

where UB is the set of nonbasic variables at their upper bounds.

To distinquish between current and new values, let $(F_B)_i$ denote the current value of the ith basic variable, and F_i denote its new value as $(F_R)_q$ changes.

As $(F_R)_q$ increases (decreases), each element in F_i will change:

1. If α_{iq} is positive, F_i will decrease (increase)
2. If α_{iq} is negative, F_i will increase (decrease).

We need to consider three possible events:

a) A basic variable may reach its lower bound (zero) first
b) A basic variable may reach its upper bound first
c) A nonbasic variable may reach the other end of its range before (a) or (b) occur.

Suppose $(F_R)_q$ is increased from zero.

For case (a) we have

$$(F_B)_i - \alpha_{iq}(F_R)_q \geq 0 \tag{3.24}$$

or

$$(F_R)_q \leq \min_{i|\alpha_{iq}>0} \{(F_B)_i / \alpha_{iq}\} \ . \tag{3.25}$$

For case (b) we have

$$(F_B)_i - \alpha_{iq}(F_R)_q \leq C_i \tag{3.26}$$

or

$$(F_R)_q \leq \min_{i|\alpha_{iq}<0} \{(C_i - (F_B)_i)/ -\alpha_{iq}\} \ . \tag{3.27}$$

For case (c) we have

$$(F_R)_q \leq C_q. \tag{3.28}$$

Thus $(F_R)_q$ must take on the smallest value of (3.25),(3.27), or (3.28). Equivalently, we can write

$$(F_R)_q = \min \{ \min_{i|\alpha_{iq}>0} \{(F_B)_i / \alpha_{iq}\}, \min_{i|\alpha_{iq}<0} \{(C_i - (F_B)_i)/ -\alpha_{iq}\}, C_q\}. \tag{3.29}$$

We must also consider the case where $(F_R)_q$ is decreased from C_q. The current values of the basis variables are given by

$$(F_B)_i = \beta_i - \sum_{j \in UB,\, j \neq q} \alpha_{ij}(F_R)_j - \alpha_{iq}C_q. \tag{3.30}$$

Thus for any new value of $(F_R)_q$, we have

$$F_i = (F_B)_i + \alpha_{iq}(C_q - (F_R)_q). \tag{3.31}$$

If we define the decrease in the nonbasic variable by $\delta = C_k - (F_R)_q$, then for case (a) we have

$$(F_B)_i + \alpha_{iq}(\delta) \geq 0 \tag{3.32}$$

or

$$\delta \leq \min_{i|\alpha_{iq}<0} \{(F_B)_i / -\alpha_{iq}\} . \tag{3.33}$$

For case (b) we have

$$(F_B)_i + \alpha_{iq}(\delta) \leq C_i \tag{3.34}$$

or

$$\delta \leq \min_{i|\alpha_{iq}>0} \{(C_i - (F_B)_i) / \alpha_{iq}\} . \tag{3.35}$$

For case (c) we have

$$\delta \leq C_q. \tag{3.36}$$

Combining the results of equations (3.33), (3.35), and (3.36) we have

$$\delta = \min \{ \min_{i|\alpha_{iq}<0} \{(F_B)_i / -\alpha_{iq}\}, \min_{i|\alpha_{iq}>0} \{(C_i - (F_B)_i)/\alpha_{iq}\}, C_q\} \tag{3.37}$$

and $(F_R)_q = C_q - \delta$. We should note if (c) occurs no change of the basis is required. However, we need to record the fact that $(F_R)_q$

has gone to the other end of its range. The new basic variables F_B' written in terms of the old basic variables F_B are

$$F_B' = F_B - \alpha_q(\pm C_q)$$

and the appropriate sign is: $\begin{cases} + \text{ if } (F_R)_q \text{ increased from zero} \\ - \text{ if } (F_R)_q \text{ descreased from } C_q. \end{cases}$

If (a) or (b) occurs, for example if $i = p$, F_B is adjusted according to

$$F_B' = \begin{cases} (F_B)_i - \alpha_{iq}(F_R)_q & i \neq p;\ (F_R)_q \text{ increased from zero} \\ (F_B)_i + \alpha_{iq}(\delta) & i \neq p;\ (F_R)_q \text{ decreased from } C_q \\ (F_R)_q & i = p. \end{cases}$$

At this point a non-basic variable has a positive value and a basic variable has a value of zero or is at its upper bound. Recall that in the partitioned equation (3.7), $(F_R)_p$ corresponds to the pth column of B. The pivot operation is completed when the tableau is rearranged so $(F_R)_q$ is basic and $(F_B)_p$ is non-basic. This means redefining the partition of A so a_q replaces the pth column of B corresponding to $(F_R)_q$ entering the basis and $(F_R)_p$ leaving the basis. Note the fact that a record must be kept of which nonbasic variables are at their upper bounds.

3.6 Steps of the Simplex Method

The procedures outlined in sections 3.2 through 3.5 form the building blocks of the bounded variable simplex method. While there exist numerous variants of this method, most computer codes use the revised simplex method. Thus in all large scale applications all the elements of $-B^{-1}R$ of the transformed problem are never stored or computed. We only need two columns of the transformed problem ($\beta = B^{-1}b$ and $\alpha_q = B^{-1}a_q$). We assume we have a basic feasible solution with some representation of B^{-1} and the current right-hand side $\beta = B^{-1}b$. The steps of the simplex are as follows:

STEP 1: Produce a pricing vector

$$\pi^T = H_B^T B^{-1}. \tag{3.38a}$$

STEP 2: Price out the nonbasic columns

a) Evaluate $z_j = \pi^T a_j$ (3.38b)
b) Evaluate $d_j = z_j - h_j$. (3.38c)

STEP 3: Select the entering nonbasic variable by choosing

a) The most positive d_j among the nonbasic variables at zero

b) The most negative d_j among the nonbasic variables at upper bounds

c) If none exists, stop; otherwise choose q corresponding to the reduced cost of the largest magnitude of a) or b).

STEP 4: Update the entering column by evaluating

$$\alpha_q = B^{-1} a_q . \tag{3.38d}$$

STEP 5: Find the leaving basic variable by selecting

$$(F_R)_q = \min \{ \min_{i \mid \alpha_{iq} > 0} \{(F_B)_i / \alpha_{iq}\}, \min_{i \mid \alpha_{iq} < 0} \{(C_i - (F_B)_i) / -\alpha_{iq}\}, C_q\} \tag{3.38e}$$

or

$$\delta = \min \{ \min_{i \mid \alpha_{iq} < 0} \{(F_B)_i / -\alpha_{iq}\}, \min_{i \mid \alpha_{iq} > 0} \{(C_i - (F_B)_i) / \alpha_{iq}\}, C_q\}. \tag{3.38f}$$

a) If none exists, stop with an unbounded solution

b) Otherwise let the minimum correspond to $i = p$.

STEP 6: Pivot by adjusting F_B so

$$F_B' = \begin{cases} (F_B)_i - \alpha_{iq}(F_R)_q & i \neq p;\ (F_R)_q \text{ increased from zero} \\ (F_B)_i - \alpha_{iq}(\delta) & i \neq p;\ (F_R)_q \text{ decreased from } C_q \\ (F_R)_q & i = p \end{cases} \tag{3.38g}$$

and a_q replaces the pth column of B.

Return to step 1.

3.7 The Dual-Simplex Method

This procedure applies the simplex method to the dual problem while working with the primal tableau. Suppose we have the standard primal tableau expressed in the form

$$\text{Min } Z = H_B^T\beta - d_1(F_R)_1 - \ldots - d_q(F_R)_q - \ldots - d_{n-m}(F_R)_{n-m}$$

$$\text{S.T.} \quad F_B = \beta - \alpha_1(F_R)_1 - \dots - \alpha_q(F_R)_q - \dots - \alpha_{n-m}(F_R)_{n-m}. \tag{3.39}$$

For ease of exposition this explanation describes the dual-simplex method for a linear program without bounds. We assume the tableau is dual feasible (optimal) but not necessarily primal feasible. That is, all the reduced costs are less than or equal to zero ($d_j \leq 0$), but one or more of the elements of the vector b is negative. Then the steps of the dual-simplex method are:

STEP 1: Choose the pivot row p by selecting

$$\beta_p = \min_{i \mid \beta_i < 0} \{ \beta_i \} . \tag{3.40}$$

a) If none exists, stop with optimal and feasible solution
b) Otherwise let the minimum correspond to i=p, say $(F_B)_p$ is the leaving basic variable.

STEP 2: Choose the entering nonbasic variable.

We must note that the reduced cost of $(F_B)_q$ in the new tableau if $(F_R)_j$ replaces $(F_R)_q$ in the basis is given by

$$\bar{d}_j = -d_j \,/\, (\alpha_j)_p \,. \tag{3.41}$$

The ratio test now becomes

$$\Theta = \min_{j \mid (\alpha_j)_p < 0} \{-d_j \,/\, (\alpha_j)_p\} \tag{3.42}$$

for $j = q$. Noting the fact that if any higher ratio were used, say for $j=r$ and $(F_R)_r$ entered the basis, and if $(-d_r \,/\, (\alpha_r)_p) > (-d_q \,/\, (\alpha_q)_p)$, then

$$\bar{d}_q = d_q - (\,(\alpha_q)_p \,/\, (\alpha_r)_p)\, d_r < 0 \tag{3.34}$$

and optimality would be lost. Thus when $(F_R)_q$ enters the basis

$$\bar{d}_j = d_j + \Theta\, (\alpha_j)_p \quad j \neq q. \tag{3.44}$$

STEP 3: Check for infeasibility

If there is no element $(\alpha_j)_p < 0$ for any of the nonbasic variables $(F_R)_j$, then no nonnegative value of $(F_R)_j$ can cause $(F_B)_p$ to be feasible;

Therefore, stop. The problem is infeasible.

Otherwise go to step 4.

STEP 4: Perform the pivot operation

As in the primal method, exchange $(F_R)_q$ with $(F_B)_q$ in the basis and return to step 1.

There are two major motivations for using the dual-simplex method.

a) Adding extra rows to the primal problem. Using the dual-simplex method this is achieved with essentially the same ease as adding an extra column to the primal and pricing out the new column with the current basis.

b) It is also used in parametric programming of the right-hand side vector.

3.8 Perturbation of the Right-Hand Side Vector C

During the development of parametric programming, most of the research effort has been spent on analyzing the problem

$$\text{Min } Z = H^T F \tag{3.45}$$

$$\text{S.T.} \quad AF = b + \Theta \hat{b} \tag{3.46}$$

$$\text{in the range} \quad 0 < \Theta \leq \Theta_{max} \tag{3.47}$$

$$F \geq 0. \tag{3.48}$$

However little analysis has been spent on the development of the parametric analysis for the problem

$$\text{Min } Z = H^T F \tag{3.49}$$

$$\text{S.T.} \quad AF = b \tag{3.50}$$

$$\underline{C} \leq F \leq C + \lambda \hat{C} \tag{3.51}$$

$$0 < \lambda < \lambda_{max}$$

where we parametrically vary the upper bounds of the variable F, with λ being a scalar.

A capacitated generalized network is a special case of problem (3.49). Therefore, in order to do parametric analysis on the capacitated generalized network, we must first analyze parametric changes in the implicit right-hand side constraints.

Thus suppose the vector C is replaced by $C + \lambda \hat{C}$, $\lambda \geq 0$. This means the capacity constraints are changed along the vector $\hat{C}$. Since the right-hand side of the primal problem is the objective of the dual problem, changing the implicit capacity constraints could be analyzed as changing the objective function of the dual problem. However, we shall now discuss the effects of changing the capacity constraints directly in the primal problem.

Recalling from section 3.2, we can partition matrix A into (B,R). We denote f_{ji} corresponding to the ith column of B by F_{Bi} and let R' be the set of indices of the columns of R. Let

$$\alpha_j = B^{-1}a_j \quad j \in R'$$

and

$$\beta = B^{-1}b \ .$$

We can write equation (3.50) in solved form as

$$F_{Bi} = \beta_i + \sum_{j \in R'} \alpha_j(-f_j) \quad i=1,\ldots,m \ . \tag{3.52}$$

If we define

$$LB = \{j \in R' | f_j \text{ is at } \underline{c}_j\}$$
$$UB = \{j \in R' | f_j \text{ is at } c_j + \lambda \hat{c}_j \}$$
$$Z = \{j \in R' | f_j \text{ is at } 0\}$$

then we can write (3.52) as

$$F_{Bi} = B^{-1} \left[b + \sum_{j \varepsilon LB} \alpha_j(-\underline{c}_j) + \sum_{j\varepsilon\ UB} \alpha_j(-\lambda \hat{c}_j - c_j) + \sum_{j\varepsilon\ Z} \alpha_j f_j \right] . \tag{3.53}$$

If we let

$$\rho_o = B^{-1} \left[- \sum_{j\varepsilon UB} \alpha_j \hat{c}_j \right]$$

and let

$$\sigma_o = B^{-1} \left[b - \sum_{j \varepsilon LB} \alpha_j \underline{c}_j - \sum_{j\varepsilon\ UB} \alpha_j c_j \right]$$

then we can rewrite (3.53) as

$$F_{Bi} = \sigma_{io} + \rho_{io}\lambda + \sum_{j\varepsilon\ Z} \alpha_j f_j \quad i=1,\ldots,m . \tag{3.54}$$

Now we consider varying λ to satisfy

$$\underline{c}_i \le \sigma_{io} + \rho_{io}\lambda \le c_i + \lambda\ \hat{c}_i \qquad i=1,\ldots,m$$

where LB_i, UB_i are the bounds on the basic variable F_{Bi}.

From (3.54) we require F_{Bi} to satisfy

$$0 \le (\sigma_{io} - \underline{c}_i) + (\rho_{io})\lambda \quad i=1,\ldots,m \tag{3.55}$$

and

$$(\sigma_{io} - c_i) + (\rho_{io} - \hat{c}_i)\lambda \leq 0 \quad i=1,\ldots,m \,. \qquad (3.56)$$

In general we consider increasing λ from zero to some upper limit $\hat{\lambda}$. We define

$$Q_1 = \{i \mid (\sigma_{io} - \underline{c}_i) \geq 0, \rho_{io} < 0\}$$
$$Q_2 = \{i \mid (\sigma_{io} - c_i) \leq 0, (\rho_{io} - \hat{c}_i) > 0\} \,. \qquad (3.57)$$

As λ is increased, (3.55) may be violated if and only if $i \varepsilon Q_1$ and (3.56) may be violated if and only if $i \varepsilon Q_2$. Thus let

$$\lambda_1 = \min_{i \varepsilon Q_1} \{(\sigma_{io} - \underline{c}_i) / -\rho_{io}\}$$

$$\lambda_2 = \min_{i \varepsilon Q_2} \{-(\sigma_{io} - c_i) / (\rho_{io} - \hat{c}_i)\}$$

and

$$\bar{\lambda} = \min(\lambda_1, \lambda_2). \qquad (3.58)$$

If the minimum is attained in (3.58) for λ_k, let $p = p_k$. Thus F_{Bp} will leave the basis at its lower bound (k=1) or its upper bound (k=2). This leads to two cases.

CASE 1: $\bar{\lambda} = \lambda_1$. In this case F_{Bp} leaves the basis by going to its lower bound. If we were to increase $\bar{\lambda}$ further, it must be true from

(3.52), we must either

(a) increase a nonbasic variable which has $\alpha_{ij} < 0$ and $j \in LB \cup Z$

or

(b) decrease a nonbasic variable which has $\alpha_{ij} > 0$ and $j \in UB \cup Z$.

If such a variable can be found, say F_{Rq}, perform a simplex pivot and make F_{Rq} basic while F_{Bp} enters the set of nonbasic variables at their lower bound.

CASE 2: $\bar{\lambda} = \lambda_2$. In this case F_{Bp} leaves the basis by going to its upper bound. If we increase $\bar{\lambda}$ further, it also must be true from (3.52) that we must either

(a) increase a nonbasic variable which has $\alpha_{pj} > 0$ and $j \in LB \cup Z$

or

(b) decrease a nonbasic variable which has $\alpha_{pj} < 0$ and $j \in UB \cup Z$.

Again if such a variable can be found, say F_{Rq}, make the appropriate simplex pivot by letting F_{Rq} become basic while F_{Bp} enters the set of nonbasic variables at their upper bound.

If no element $B^{-1}r_{pq}$ of the appropriate sign can be found in the "blocking row" p at any step, the algorithm terminates.

An algorithm to accomplish the above procedure would have the following general steps:

STEP 0: Solve the original problem to optimality.
Set $i = 1, \lambda_{i-1} = 0$, and $\lambda_i = 0$.

STEP 1: Determine the element $B^{-1}r_{pq}$ of the appropriate sign. If none can be found, stop; the current basis is optimal for all values of $\lambda > \lambda_i$ or any further iterations will cause the problem to become infeasible. Otherwise calculate $\bar{\lambda}$.

STEP 2: Let $\lambda_{i-1} \leftarrow \lambda_i$ and $\lambda_i \leftarrow \bar{\lambda}$. For $\lambda \varepsilon$ $[\lambda_{i-1}, \lambda]$ the current basis will remain optimal. Remove F_{Bp} from the basis and pivot F_{Rq} into the basis. Update the elements of the simplex tableau and the objective function. Return to step 1.

3.9 Perturbation of the Right-Hand Side Vector b

If in equation (3.50) we replace b by $b + \lambda \hat{b}$, we note the term $H_B B^{-1} R - H_R$ will not be affected, that is dual feasibility will not be affected. The only change is that $B^{-1} b$ will be replaced by $B^{-1}(b + \lambda \hat{b})$ and accordingly the objective function becomes

$$H_B B^{-1}(b + \lambda \hat{b}).$$

As long as $H_B B_{-1}(b + \lambda \hat{b})$ remains nonnegative, the current basis will remain optimal. We can, therefore, determine the value of λ at which another basis becomes optimal. Let $S = \{i \mid B^{-1} b_i < 0\}$. If $S = \Phi$, then the current basis is optimal for all values $\lambda \geq 0$. Otherwise, as before let

$$\hat{\lambda} = \min_{i \in S} \{B^{-1} b_i \, / \, -B^{-1} \hat{b}_i\} \; .$$

Let $\lambda_1 = \hat{\lambda}$. For $\lambda \in [\, 0, \lambda_1 \,]$, the current basis is optimal, where

$$F_B = B^{-1}(b + \lambda \hat{b}).$$

At λ_1 the right-hand side is replaced by $B^{-1}(b + \lambda \hat{b})$, F_{Bq} is removed from the basis, and an appropriate variable, according to the dual-simplex criteria, enters the basis. The process is repeated to find the new range [λ_1, λ_2] over which the new basis is optimal [$\hat{\lambda} = \lambda_2$]. We terminate the process when S is empty, in which case the current basis is optimal for all values of λ greater than or equal to the last value of λ, or else when all the entries in the row whose right-hand side dropped to zero are nonnegative. In this case no feasible solutions exist for all values λ greater than the current value.

CHAPTER 4

BACKGROUND IN NETWORK FLOW PROGRAMMING

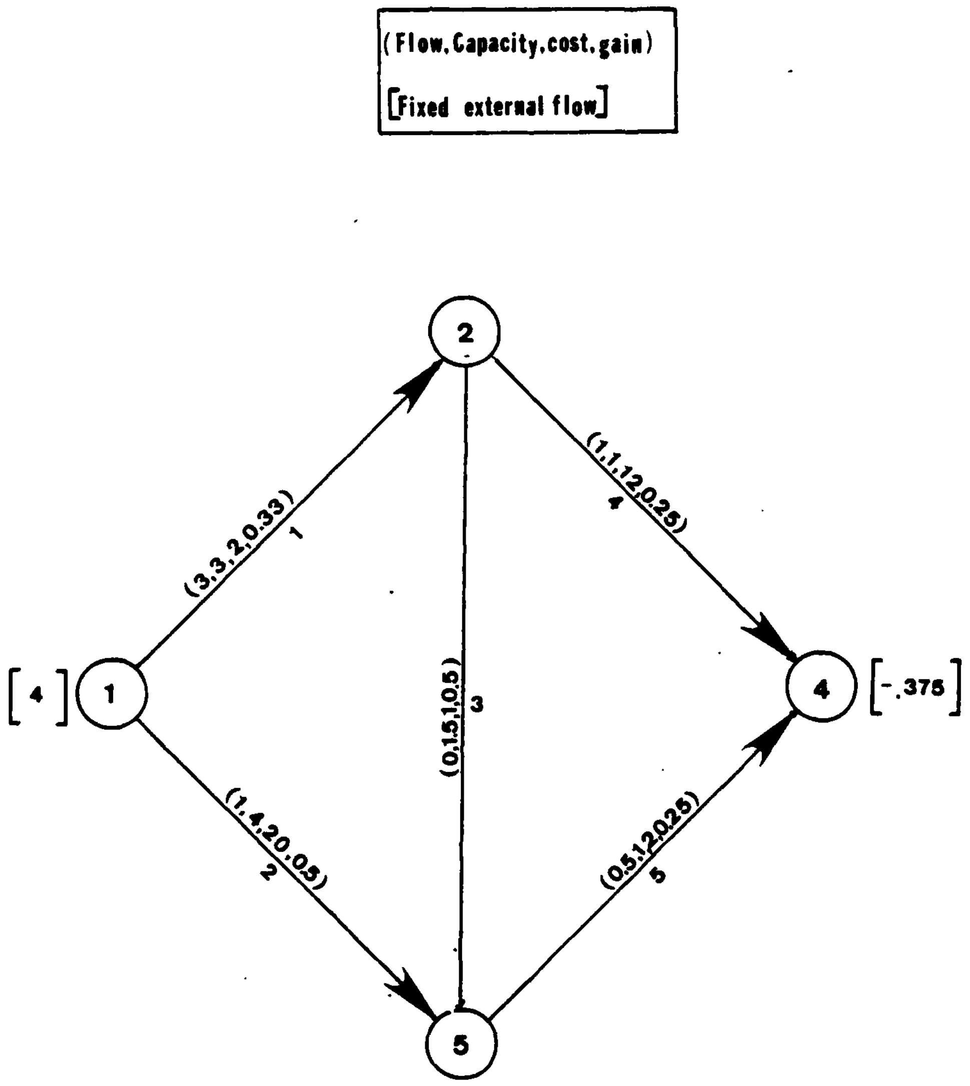
4.1 Introduction

To better understand the solution method to be developed in Chapter 5, we shall first discuss some of the concepts for the solution of a generalized minimum cost flow problem. A detailed explanation of the concepts discussed below can be found in the text by Jensen and Barnes (46).

To illustrate a typical generalized minimum cost flow problem, consider the network model depicted in Figure 4.1. The algebraic representation of the model is

$$\begin{aligned} \min \quad & H\,F \\ \text{S.T.} \quad & A\,F = b \\ & F \leq C \\ & F \geq 0 \end{aligned}$$

where



Example Generalized Network with Optimal Solution
Figure 4.1

$H = (2,20,1,12,2)$

$F^T = (3,1,0,1,0.5)$

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ -.33 & 0 & 1 & 1 & 0 \\ 0 & -.5 & .5 & 0 & 1 \\ 0 & 0 & 0 & -.25 & -.25 \end{bmatrix}$$

$B^T = (4,0,0,-.375)$

$C^T = (3,4,1.5,1,1.2)$.

4.2 Network Structure

The structure of a network model is defined by nodes and arcs. A node i is an element of the set of nodes, N = [1,2,. . .,i,. . .,n]. An arc may be defined by an ordered pair of nodes (i,j) or as an element, say k, of the set of arcs M = [1,2,. . .,k,. . .,m]. Thus an arc may be identified as an arc k, arc k(i,j), or (i,j). An arc k(i,j) is said to originate at node i and terminate at node j. Alternatively, i is called the origin node and j is called the terminal node of arc k. We identify the origin and terminal lists

$$O = [\ o_1,\ o_2, \ldots,\ o_m\ ]$$
$$T = [\ t_1,\ t_2, \ldots,\ t_m\ ]$$

where o_k and t_k are the origin and terminal nodes, respectively, of arc k. The collection of nodes and arcs is a directed network written D = [N,M], where the values n and m and the vectors O and T are sufficient to completely define the network.

A variable quantity that characterizes most of the problems to be considered is arc flow. Written as f_k, arc flow usually models some physical quantity such as the flow of fluid, flow of people, or flow of money. Let f_k be the flow in arc k at its origin and f_k' is the flow in the arc at its terminal node. In a generalized network, flow is not conserved in the arc. With a nonzero gain parameter, a_k,

$$f_k' = a_k f_k$$

If $a_k = 1$ the flow is conserved; if $a_k < 1$, flow is lost; and if $a_k > 1$ flow is gained in the arc. The flow vector is defined as

$$F^T = [\ f_1,\ f_2, \ldots, f_m\ ]$$

where F^T indicates the transposed F, that is a column vector.

Associated with each flow in an arc is a cost. The arc cost is a function of the flow and is written

$$h_k(f_k) = h_k f_k$$

where h_k is an arc parameter giving the arc cost per unit flow. The cost vector is a row vector, written as

$$H = [\ h_1,\ h_2, \ldots, h_m\ ].$$

The arc cost is a function of only the flow in the arc and not a function of flow in the other arcs. The network cost is the sum of the arc costs. Thus

$$HF = \sum_{k=1}^{m} h_k f_k.$$

The arc capacity c_k is an upper bound for the flow on each arc. The arc capacity implies the constraint:

$$f_k \leq c_k.$$

The arc capacity vector is a column vector, written as

$$C^T = [\ c_1,\ c_2,\ldots,c_m\ ].$$

In many applications a common requirement is that the arc flow be at least as great as some lower bound. We therefore provide the lower bound $\underline{c}_k$ as an input parameter for each arc to constrain the flow f_k such that

$$f_k \geq \underline{c}_k.$$

The lower bound vector is also a column vector and is written as

$$\underline{C}^T = [\ \underline{c}_1,\ \underline{c}_2,\ldots,\underline{c}_m\ ].$$

A gain is the parameter that models a linear increase or decrease in flow as it passes through an arc. If $a_k = 1$, we have no change in the flow as it passes through the arc. If $a_k = 1$, for all k, we have a pure network problem. When $0 < a_k < 1$, flow is decreased as it passes through the arc. When $a_k > 1$, flow is increased. It is possible to have negative gains, but the physical interpretation is not obvious. Zero gains are not allowed.

External flows represent connections to the world outside the system being modeled. There are two kinds of external flows for a node: the fixed flow and the slack flow. The fixed flow at node i, b_i, enters the network if $b_i > 0$ and leaves the network if $b_i < 0$. Any feasible solution stipulates network flows such that all fixed external flows are satisfied. The slack external flow is a variable quantity which ranges from zero to some upper bound defined for the node. The value of the slack external flow is to be determined by the solution procedure. An external slack input is represented by an arc to the node from a specially designated node called the slack node. An external slack output is represented by an arc from the node to the slack node. The slack node is different from all other nodes in that the slack node has the capability to absorb or generate any required quantity of flow.

A feasible flow F conserves flow at all nodes of the network except the slack node. With slack arcs defined as above, all flows except fixed external flows are arc flows. Conservation of flow for each node implies that the total arc flow leaving a node minus the total arc flow entering the node must equal the fixed external flow at that node. Algebraically, the conservation-of-flow constraint for a general node i is

$$\sum_{k \in M_{Oi}} f_k - \sum_{k \in M_{Ti}} a_k f_k = b_i .$$

4.3 A Useful Simplification

It is possible to express any network flow problem with nonzero arc lower bounds as an equivalent problem with all zero lower bounds. Therefore, we delete the lower bound from the parameter list by making the following transformation for each arc k(i,j) with nonzero lower bound $\underline{c}_k$:

a) replace $\underline{c}_k$ by zero

b) replace c_k by $c'_k = c_k - \underline{c}_k$

c) replace f_k by $f'_k = f_k - \underline{c}_k$

d) replace b_i by $b'_i = b_i - \underline{c}_k$

e) replace b_j by $b'_j = b_j + a_k\underline{c}_k$.

This transformation is performed one arc at a time for all arcs with nonzero lower bounds. Each new transformation uses the updated parameter values that resulted from the previous step.

4.4 Expanded Networks

It is convenient from the viewpoint of obtaining optimal solutions to the generalized network flow programming problems to define an expanded network which may be obtained directly from the original network.

We define the mirror arc, $-k$, for each $k \in M$ such that arc $-k$ connects the same nodes as arc k, but has opposite direction.

$$o(-k) = t(k)$$
$$t(-k) = o(k).$$

The expanded network, $D_E = [\ N, M_E\ ]$, has the same node set as $D = [\ N,M\ ]$, but its arc set contains not only the arcs of D, but also all of the mirror arcs as well. That is, if $M = [1,2,\ldots,m]$ then $M_E = [1,2,\ldots,m,-1,-2,\ldots,-m]$.

If the expanded network parameters are denoted with asterisks, their general definition may be written as

Forward arcs: $h_k^* = h_k$
$a_k^* = a_k$

Mirror arcs: $h_{-k}^{*} = -h_k / a_k$

$a_{-k}^{*} = 1/ a_k$.

Flows in the expanded network imply changes in the flows of the original network by the relation

$$\Delta_k = f_k^{*} - (f_{-k}^{*})(a_{-k}^{*}) = f^{*} - f_{-k}^{*} / a_k$$

where Δ_k is the change in flow in the original network. The procedures will assure that a forward and mirror arc will never simultaneously have nonzero flow.

4.5 The Marginal Network

The marginal network $D^{*} = [N,M^{*}]$ has the same node set as D and D_E. The arc set D^{*} consists of that subset of M_E that are admissible arcs. A forward arc is admissible if the flow is not already equal to the arc capacity, that is if the flow can still be increased on that arc in the original network. A mirror arc is admissible if the flow on the corresponding forward arc is not equal to zero; that is if the flow can be decreased on the associated arc in the original network.

4.6 Basis and Basic Solutions

The basis for a generalized minimum cost flow problem will consist of n-1 arcs chosen so that the associated n-1 columns are linearly independent. For a pure network problem, such a selection forms a directed spanning tree. For the generalized problem, a selection of n-1 linearly independent columns may result in a more general form. In particular, a basis for a problem may contain one or more cycles.

To illustrate these concepts consider the network of Figure 4.1. In this figure no slack node has been explicitly defined. For this example we arbitrarily choose node 1 as the slack node. The constraint matrix formed by the conservation-of-flow equations is

$$\begin{array}{c} \text{ARCS} \\ \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \end{array}$$

$$A = \left[\begin{array}{ccccc} -a_1 & 0 & 1 & 1 & 0 \\ 0 & -a_2 & -a_3 & 0 & 1 \\ 0 & 0 & 0 & -a_4 & -a_5 \end{array}\right] \begin{array}{l} 2 \quad \text{NODES} \\ 3 \\ 4 \end{array} .$$

Note that the constraint associated with the slack node has been deleted. This constraint is redundant and must be deleted to obtain a set of independent rows. Our problem is to choose a set of three independent columns from this matrix. The set will be independent if the square matrix formed by the columns has a nonzero determinant. For example, if we choose the columns whose associated arcs form a tree, arcs 1, 3, and 4, the basis matrix becomes

$$B = \begin{bmatrix} -a_1 & 1 & 1 \\ 0 & -a_3 & 0 \\ 0 & 0 & -a_4 \end{bmatrix} .$$

The determinant of the basis matrix is

$$|B| = -a_1 a_3 a_4 \quad .$$

Since the gain factors are nonzero, this determinant can never equal zero. It can be shown that for a generalized network, any selection of columns that describes a tree can be arranged to form a matrix with diagonal elements equal to the gain factors and the lower off-diagonal elements equal to zero. Thus $|B|$, the determinant, can always be written as

$$|B| = \pm \left(\prod_{k \in M_B} a_k \right)$$

where M_B is the set of arcs in the basis tree.

In order to form a tree in the manner just shown, one of the arcs chosen must originate at the slack node. Otherwise it is impossible to arrange the arcs so that the first column has only one nonzero element. Pictorially, we will represent such a basis as a directed tree rooted at the slack node. Where it is necessary, we will use mirror arcs to obtain the directed tree. Figure 4.2 illustrates two basis trees for the example problem.

Next, consider the case in which the arcs chosen form a cycle. For example, select arcs 3, 4, 5 from Figure 4.1, so that

$$B = \begin{bmatrix} 1 & 1 & 0 \\ -a_3 & 0 & 1 \\ 0 & -a_4 & -a_5 \end{bmatrix}$$

and

$$|B| = a_4 - a_3a_5.$$

If $B \neq 0$ then the cycle is an acceptable basis. We note that the cycle forms a basis if and only if

$$a_3 a_5 \neq a_4$$

or

$$a_3 a_5 / a_4 \neq 1.$$

This has a graphical interpretation which is illustrated in Figure 4.2. Starting at node 2 and passing around the cycle, we encounter arcs 3, 5, -4. The gain of the cycle is then defined as the product of the gains that form a cycle. We assign the symbol β to the cycle gain. Thus for the example

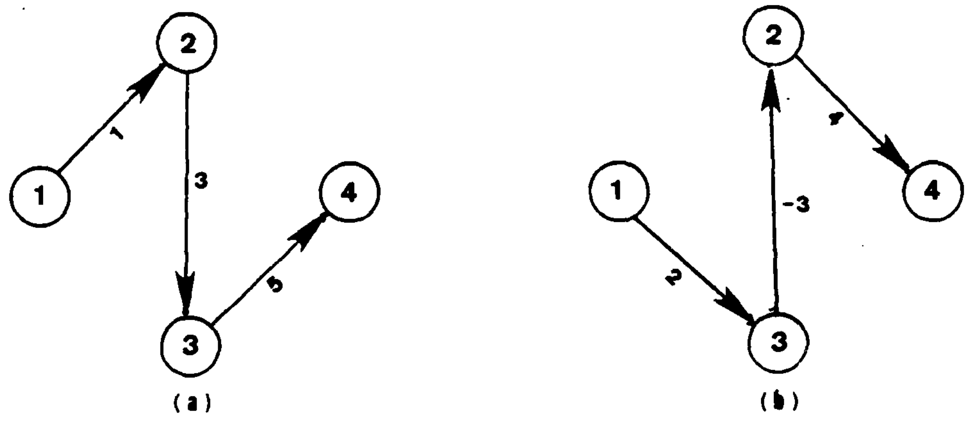
$$\beta = a_3 a_5 / a_4$$

and this cycle would be an acceptable basis if β does not equal one.

In general, it can be shown that for a basis component with arcs M_B that includes cycle arcs M_C ($M_C \subset M_B$), the determinant of the associated matrix will be proportional to

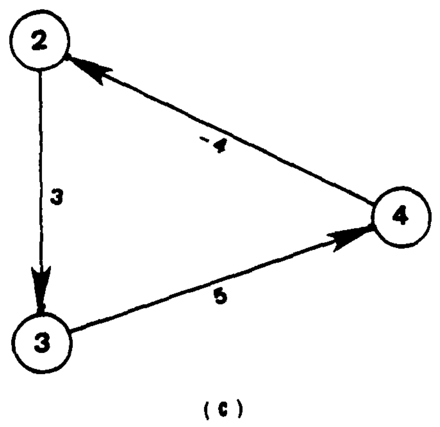
$$\prod_{k \in M_B} a_k - \prod_{k \in M_B - M_C} a_k \quad .$$

We represent a basis component that includes a cycle as a directed graph with all noncycle arcs directed as if the cycle were the root. It is convenient to think of these arcs as a tree rooted at a cycle. A particular basis may be a combination of components as illustrated



(a) and (b) are spanning trees

(c) is a network with a cycle



Example of Basis Networks
Figure 4.2

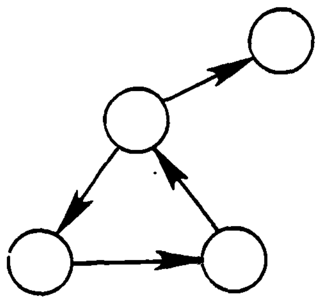
in Figure 4.4.

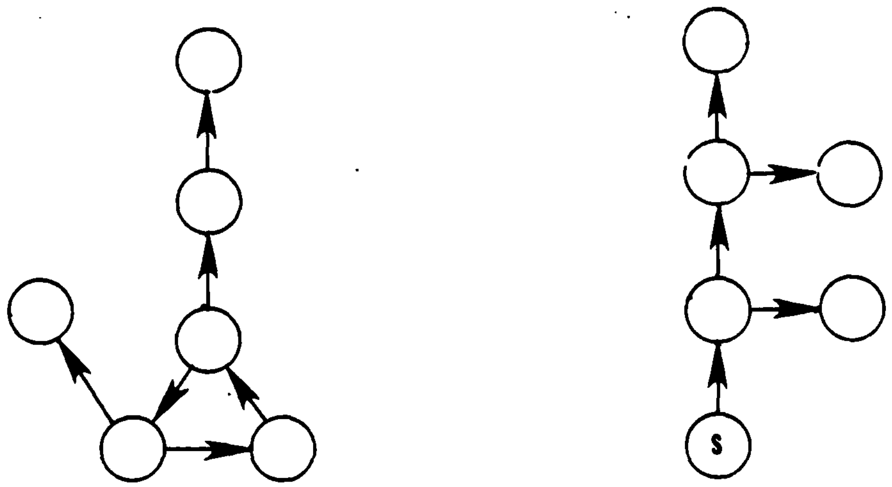
4.7 Pointer Representation of the Basis

To represent the components of a tree, three labels are assigned to each node. For node i, the three labels would be the back pointer $P_B(i)$, the forward pointer $P_F(i)$, and the right pointer $P_R(i)$. The back pointer is the unique arc which terminates at node i. The forward pointer is the terminal node of the arc originating at node i. The right pointer is the node appearing to the right of node i in the tree. The basis can also be represented with the preorder traversal method (37). In this method the preorder traversal list P_p simply stores the pointer representation where $P_p(i)$ is set equal to the node number to which node i points. In our descriptions, as well as in Jensen and Barnes (46), the root node is always the slack node. Figure 4.5 illustrates these concepts.

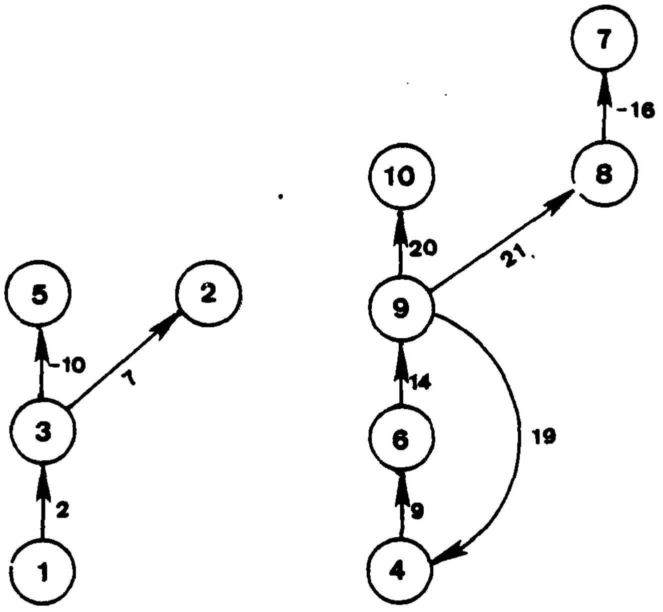
4.8 Linear Programming Results for Network Models

In order to relate material presented in Chapter 3 and to help lay the foundation for much of the material presented in Chapters 5 and 6, it seems appropriate to summarize some of the


Basis Cycle
Figure 4.3



Basis with Several Components
Figure 4.4



Node	1	2	3	4	5	6	7	8	9	10
PB(Arc)	0	7	2	19	-10	9	-16	21	14	20
PF(Node)	3	0	5	6	0	9	0	7	10	0
PF(Node)	0	0	0	0	2	0	0	4	0	8

Basis Network with a Cycle
Figure 4.5

major results of linear programming theory that pertain to network flow problems. In terms of the parameters and terms previously defined in this chapter, the generalized single commodity linear minimum cost flow problem may be presented as the linear programming primal problem:

$$\text{Min} \quad \sum_{k=1}^{m} h_k f_k \tag{4.1}$$

$$\text{S.T.} \quad \sum_{k\varepsilon\, M_{0i}} f_k - \sum_{k\varepsilon\, M_{Ti}} a_k f_k = b_i \qquad i=1,\dots,n-1 \tag{4.2}$$

$$f_k \leq c_k \qquad k=1,\dots,m \tag{4.3}$$

$$f_k \geq 0 \qquad k=1,\dots,m \tag{4.4}$$

where node n is the slack node.

The dual to the linear program, may be written as

$$\text{Min} \quad \sum_{i=1}^{n-1} \pi_i b_i + \sum_{k=1}^{m} \delta_k c_k \tag{4.5}$$

$$\text{S.T.} \quad \pi_i - a_k \pi_j + \delta_k \geq -h_k \qquad k=1,\dots,m$$

$$i=0(k),\ j=T(k) \tag{4.6}$$

$$\pi_i \text{ unrestricted} \qquad i=1,\dots,n-1 \tag{4.7}$$

$$\delta_k \geq 0\ . \tag{4.8}$$

The primal-dual conditions for an optimal solution of the linear program have been specialized to the generalized minimum cost network flow problem in the following three theorems.

THEOREM 1: Given a solution F to the primal problem and a solution $[\pi, \delta]$ to the dual problem, the solutions are optimal if and only if:

1. F is feasible for the problem; that is (4.2-4.4) are satisfied.
2. $[\pi, \delta]$ is feasible for the dual problem; that is (4.6) and (4.8) are satisfied.
3. Complementary slackness is satisfied; that is
   (a) If $f_k(i,j) > 0$ then $\pi_i - a_k \pi_j + \delta_k = -h_k$.
   (b) If $f_k < c_k$, then $\delta_k = 0$.
   (c) If $\pi_i - a_k \pi_j + \delta_k > -h_k$, then $f_k(i,j) = 0$.
   (d) If $\delta_k > 0$, then $f_k = c_k$.

Since $C \geq 0$, a restricted condition of dual feasibility may be succinctly stated as follows:

THEOREM 2: If $[\pi, \delta]$ is an optimal solution to the dual problem, then $\delta_k = \max[0, -h_k - \pi_i + a_k \pi_j]$.

Theorem 2 allows the optimality conditions for the problem to be written as follows:

THEOREM 3: Given a solution F to the primal problem and a partial solution π to the dual problem, the solutions are optimal to their respective problems if and only if the following considerations are satisfied:

1. Primal feasibility.
2. $\delta_k = \max[0, -h_k - \pi_i + a_k\pi_j]$
   (restricted dual feasibility).
3. Complementary slackness:

   (a) $\pi_i - a_k \pi_j = -h_k$ for $0 < f_k < c_k$.

   (b) $f_k = 0$ for $\pi_i - a_k \pi_j > -h_k$.

   (c) $f_k = c_k$ for $\pi_i - a_k \pi_j < -h_k$.

These results lay the foundation for various solution procedures used to solve generalized network flow problems, which will now be presented.

4.9 Solution Algorithms for Network Flow Problems

A wide variety of solution algorithms have been introduced to solve network problems of various types. In general, they are finite iterative procedures designed to obtain a solution that satisfies the conditions stated in section 4.8: primal feasibility, restricted dual feasibility, and complementary slackness. The major difference between the algorithms is the order in which the conditions are satisfied. Each of the algorithms generally takes on a different form as it is applied to various problem classes, but the steps noted below are consistently followed. While the conditions for optimality do not require that the solutions be basic, maintaining a basis will often result in a savings of computational time.

(a) Primal Approach - The primal approach iteratively derives F and π such that F is primal feasible while attempting to achieve complementary slackness. If we define e_{ck}, a measure of the violation of complementary slackness for arc k, as

$e_{ck} = \max [\, d_k f_k, (f_k - c_k) d_k \,]$

for feasible flow, e_{ck} will be positive if and only if complementary slackness is violated. Let

$$E_c = \sum_{k=1}^{m} e_{ck}.$$

When E_c is driven to zero, F is optimal.

ALGORITHM

1. Find F and π that satisfy primal feasibility.
2. Find an arc such that $e_{ck} > 0$. If there are none, stop. Otherwise go to step 3.
3. Find a new F and π that reduce e_{ck} for the arc while maintaining primal feasibility. Go to step 2.

(b) Dual Node Infeasible Algorithm - The dual node infeasible algorithm maintains complementary slackness and satisfies all primal feasibility requirements except conservation of flow. Let b_i' be the external flow requirement at node i that would satisfy conservation of internal flows at node i, under current arc flow requirements. Let $e_{Ni} = |b_i - b_i'|$ be a measure of the infeasibility for node i and let

$$E_N = \sum_{i=1}^{N-1} e_{Ni}.$$

In this approach we achieve optimality by iteratively changing F and π in such a way to ultimately force E_N to zero.

ALGORITHM

1. Find π and $F \geq 0$ that satisfy complementary slackness and the arc capacity restrictions. Let b_i' be the

node flows required to obtain conservation of flow.

2. Find a node such that $e_{Ni} \geq 0$. If there are none, stop. Otherwise go to step 3.

3. Find a new F and π that reduces the infeasibility of the node while maintaining complementary slackness and arc feasibility. Go to step 2.

c) Dual Arc Infeasible Algorithm - During each iteration of the dual arc infeasible approach, complementary slackness and conservation of flow conditions are satisfied. However, one or more arcs will have flows above their capacities or below zero. Let $e_{Ak} = \max [\ -f_k, 0, f_k - c_k\ ]$ be a measure of infeasibility of arc k. Let

$$E_A = \sum_{k=1}^{m} e_{Ak}$$

measure the infeasibility of the network.
The dual arc infeasible algorithm achieves optimality by iteratively modifying F and π to force $E_A = 0$.

ALGORITHM

1. Find an initial F and π that satisfies complementary slackness and conservation of flow.
2. Find an arc k for which $e_{Ak} > 0$. If none exist, stop. Otherwise go to step 3.
3. Modify F and π in order to reduce e_{Ak} while maintaining conservation of flow and complementary slackness. Go to step 2.

4.10 Network Manipulation Subroutines

At this point we have presented all the necessary background except for one major area. In Chapters 5 and 6, numerous references will be made to various subroutines developed by Jensen and Barnes (46). It would be impractical to recreate the development of these subroutines at this time. Instead the following material is a review of the purpose of each subroutine referenced.

a) ADDTRE- To add an arc k(i,j) to a forest of trees. Node i and j must be in different trees and node j must be the root of a tree.

b) CYCLE- To compute the gain of a cycle defined by the backpointers and the cost of circulating one unit of flow around the cycle.

c) DELTRE- To delete an arc from the pointer representation of the tree.

d) DSHRTG- To derive the shortest path from node s to node t in a generalized network when all costs are positive and all gains are $\leq$ 1.

e) DUAL- To compute the dual variables for the basis network rooted at a given node II.

f) FLOWG- To change the flow in each arc by an amount prescribed by MF and G(j). (See Chapter 5 for a definition of these quantities).

g) ORIG- To determine the list of arcs originating at node I.

h) ORIGSG- To accept an arc data item and store it in an arc list ordered by ascending origin node.

i) PSHRTG- To solve the shortest path problem for the generalized network.

j) READG- To read and store node and arc data for the generalized minimum cost flow problem.

k) ROOTG- To find the list of arcs (LISA) and the list of nodes (LISN) that are in the directed tree rooted at node IROOT.

l) TERM- To determine the list of arcs terminating at node I.

m) TERMS- To create LT, a list of arc indices in order of increasing terminal node.

n) TREINT- To construct a pointer representation of a tree given knowledge of the back pointer arcs which make up the tree.

o) TRECHG- To delete an arc k from the basis tree, insert another arc k in the basis tree, and redirect certain arcs in the tree to maintain a directed tree.

CHAPTER 5

PARAMETRIC ANALYSIS OF A GENERALIZED NETWORK

5.1 Introduction

In this chapter we will describe an algorithm to perform parametric analysis on the capacity vector and the external flow vector of a generalized network. The mathematical development of this material as related to linear programming theory can be found in Chapter 3, sections 3.8 and 3.9. We will first develop the detailed algorithms for the variation of the capacity vector and then extend this to the external flow vector.

5.2 Parametric Analysis for Arc Capacities

The parametric sensitivity analysis algorithm described in this section is applicable to the generalized minimum cost flow problem in which the signs of the arc costs and arc gains are unrestricted. The algorithm is a dual algorithm since the iterative step first selects an arc to be deleted from the basis and then selects another arc to be added to the basis. We may state the general steps of the algorithm as follows:

1. Solve the original problem to optimality.
2. Specify the list of candidate arcs and the list of parametric parameters.
3. Find the flow augmenting trail or trails which are generated by nonbasic candidate arcs . Determine the arc to leave the basis.
4. Find an arc to enter the basis.
5. Change the basis by deleting the leaving arc and inserting the entering arc. Modify the dual variables to satisfy complementary slackness for the new basis. Return to step 3.

Before stating the algorithm in detail, each step of the outline is discussed.

5.3 Initial Optimal Solution

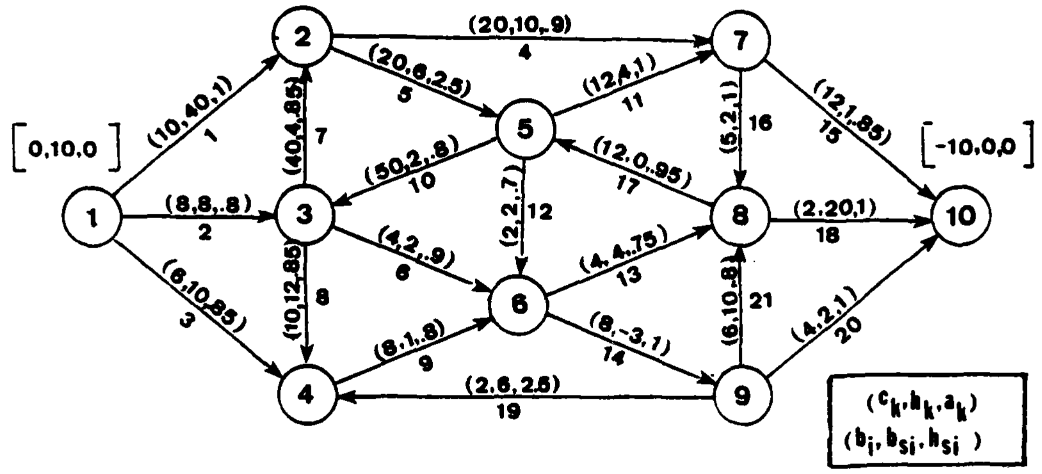
The initial problem can be solved to optimality by either a primal simplex algorithm or a dual flow augmentation algorithm for the generalized minimum cost flow problem. For details of these solution methods, which will not be repeated here, see PGAINS [46, pg 332] and INCREMG [46, pg 314].

5.4 Candidate Arc and Parametric Parameter Lists

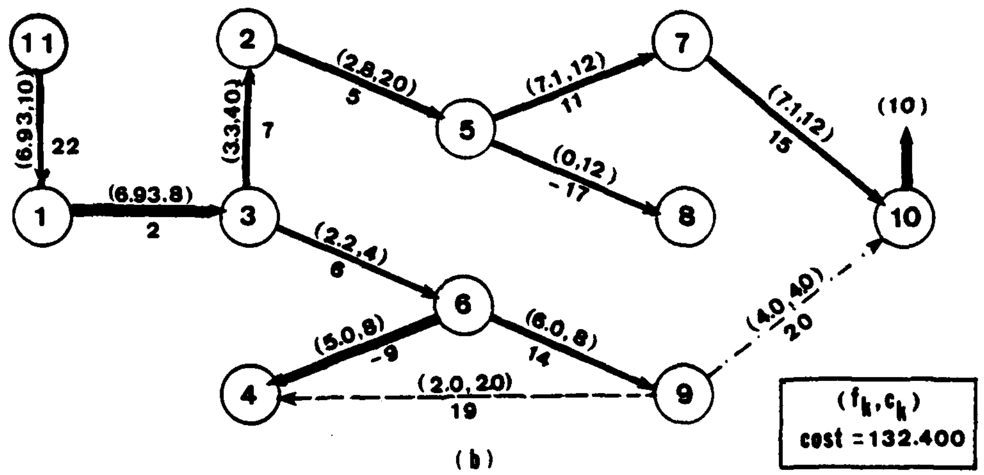
For any mathematical program, the analyst may wish to determine how the solution will vary if certain parameters are changed. As outlined in Chapter 3 section 3.8, if the vector of capacities C is changed to be $C + \lambda \hat{C}$, then a new series of optimal bases can be determined as a function of the scalar λ. We will call the arcs whose capacities are to be varied the candidate arcs and let X be the set of candidate arcs. Associated with the candidate arc set is the parametric parameter set P. The elements of P are the rates at which the capacities on the candidate arcs are to be changed.

5.5 Computing the Effects of Parametric Changes on Basic Flows

Parametric or sensitivity analysis requires that one be able to specify the effects of various parametric changes on the flows in the basic arcs. Assuming the original network problem has been solved to optimality by some procedure, a set of basic arcs is defined with flows on each basic arc. An example network problem is shown in Figure 5.1(a) with the optimal solution shown in Figure 5.1(b). Basic arcs are shown with heavy lines. Nonbasic arcs at their lower bound (zero) are not shown. Nonbasic arcs at capacity



(a)



(b)

Generalized Network Flow Problem with Optimal Solution
Figure 5.1

(c_k) are shown as dashed lines. The following sections describe how the marginal flow change in each basic arc is determined for various changes in the network parameters. Throughout this discussion, the networks of Figures 5.1, 5.2, and 5.3 are used as examples.

5.5.1 Flow Change at a Node

If the external flow changes at a node, the marginal changes in the basic flows can be predicted by the procedures of this section. The same procedures will be expanded in the next section to find the marginal flow changes caused by capacity changes of a nonbasic arc.

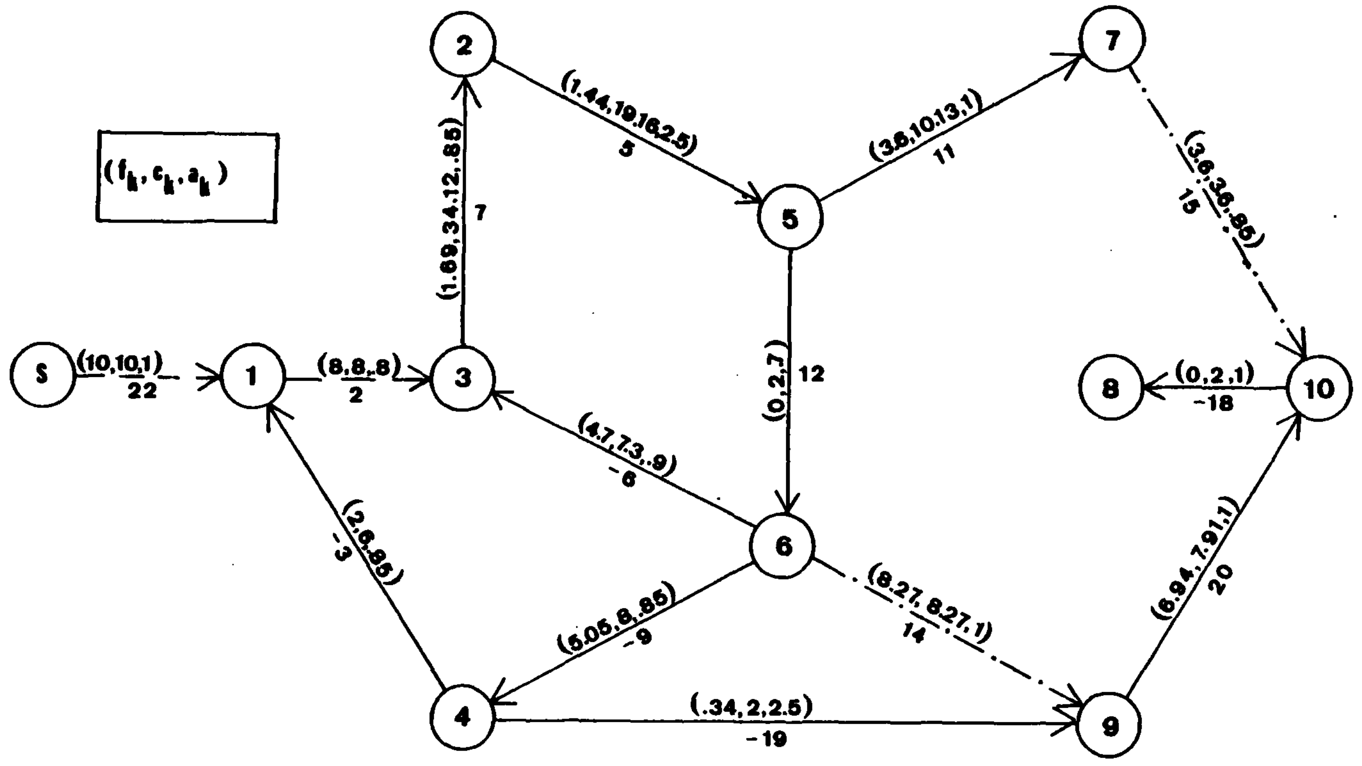
In order to develop the relationships that guide the flow changes at a node, we require some additional definitions. A flow augmenting trail is a list of basic arcs. The first arc on the trail may either originate at the slack node, node n, or at a node on a cycle. The last arc on the trail may terminate at any arbritary node v. Later we will specify exactly what this last node will be in our applications. We will define the node gain for any node u, γ_u, $u \varepsilon N$, as the inverse product of the arc gains on the trail from node u to the last node on the trail. We will denote T_u as the set of arcs on the trail from node u to the last node on the trail and $\gamma_u = 1$ if u is the last node on the trail. Once the value of γ_u is computed, where u is any node in the network, we can compute the flow in a basic arc k(i,j) as a function of the flow change at node j. Let Δ be the flow increase at node j. The corresponding increase in arc k is

$$\Delta_k = \gamma_j \Delta / a_k.$$

To illustrate the concept of flow augmenting trails, consider a trail from node 11 to node 9 in Figure 5.1. In this case the arcs on the trail are [14,6,2,22] and the nodes on the trail are [9,6,3,1,11]. Note the trail is found by tracing the backpointers from the end of the trail. We may calculate the node gains for each node on the trail as

$$\begin{aligned}
\gamma_9 &= 1 \\
\gamma_6 &= 1/a_{14} \\
\gamma_3 &= 1/(a_{14}a_6) \\
\gamma_1 &= 1/(a_{14}a_6a_2). \\
\gamma_{11} &= 1/(a_{14}a_6a_2a_{22}).
\end{aligned}$$

As an example of a trail including a cycle, consider the trail starting at node 6 and goes to node 7 in Figure 5.2. The arcs on the trail are [11,5,7,-6,12] and the nodes are [7,5,2,3,6]. Again



Optimal Generalized Network Basis
Figure 5.2

we note that the trail found by tracing backwards from node 7 by using backpointers. We may calculate the node gains for each of the nodes on the trail as

$$\gamma_7 = 1$$
$$\gamma_5 = [1/a_{11}](\beta / \beta - 1)$$
$$\gamma_2 = [1/(a_{11}a_5)](\beta / \beta - 1)$$
$$\gamma_3 = [1/(a_{11}a_5a_7)](\beta / \beta - 1)$$
$$\gamma_6 = [1/(a_{11}a_5a_7a_{-6})](\beta / \beta - 1).$$

where β is the cycle gain described in section 4.6.

Thus in general, if node u is not on a cycle we may write

$$\gamma_u = 1/ \prod_{k \in T_u} a_k \qquad (5.1)$$

and if node u is on a cycle

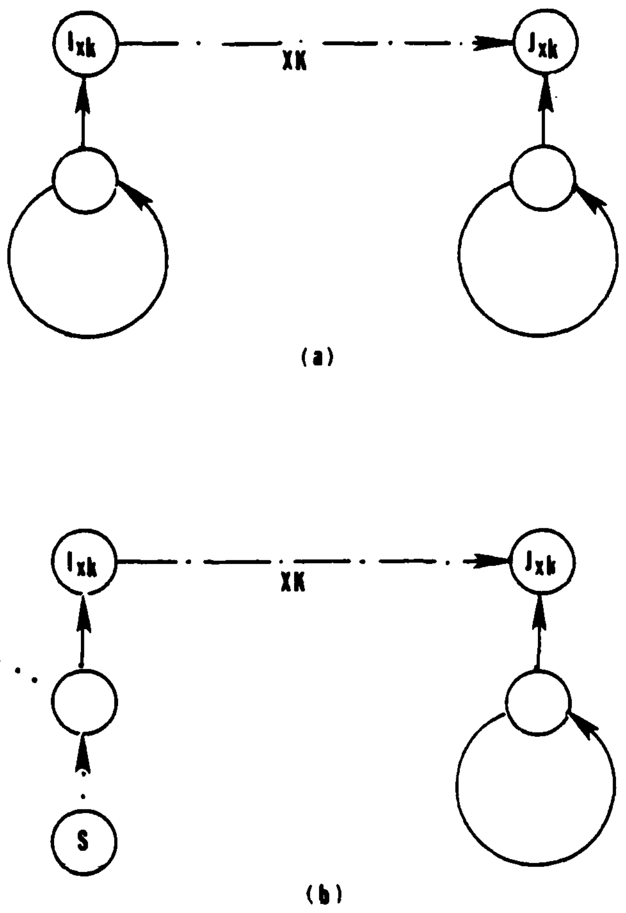
$$\gamma_u = (1/ \prod_{k \in T_u} a_k)(\beta/\beta - 1) \qquad (5.2)$$

and $\gamma_u = 1$ if u is the last node on the trail.

5.5.2 Capacity Change on a Single Nonbasic Arc

The effect of a capacity change in a nonbasic arc depends on whether the flow on this arc is zero or at capacity. If the flow is zero, there is no effect and no analysis is necessary. If the flow is at capacity, then as the capacity changes the flow in the arc must change simultaneously so that the arc flow remains at capacity. The arc will remain nonbasic with flow at its upper bound. The basic flows must change to accommodate the flow change in the nonbasic arc. Increasing or decreasing the flow in the nonbasic arc x_k has the effect of changing the node flows at its two terminal nodes.

For a nonbasic arc x_k two augmenting trails are defined. There is an augemting trail that starts at the slack node n or at a cycle and ends at the origin node of x_k. The second flow augmenting trail starts at the slack node n or at a cycle and ends at the terminal node of x_k. These two flow augmenting trails may have no nodes in common, or one or more nodes in common. There are many possible combinations of flow augmenting trails. Consider the nonbasic arc in Figure 5.3(a). In this case both flow augmenting trails originate at a cycle and end at the origin node and terminal node of x_k respectively. We should note that the trails have no nodes in common. In Figure 5.1(b), nonbasic arc 20 forms two



— · — · —➤	Candidate Arc
· · · · · · · · · ·➤	Previous Augmenting Trail

Flow Augmenting Trails
Figure 5.3

trails. One trail starts at node 11 and ends at node 9. It has arcs [14,6,2,22] and nodes [9,6,3,1,11]. The second trail begins at node 11 and ends at node 10. It has arcs [15,11,5,7,2,22] and nodes [10,7,5,2,3,1,11]. In this case the trails intersect at node 3 and have common arcs 2 and 22 and common nodes 3, 1, and 11.

In Figure 5.2 nonbasic arc 15 also forms two trails. The first trail begins at node 5 and ends at node 7. The trail contains arcs [11,5,7,-6,12] and nodes [7,5,2,3,6]. In this case the trail originates at a cycle. The second trail begins at node 6 and ends at node 10. It contains arcs [20,-19,-9,12,5,7,-6] and nodes [10,9,4,6,5,2,3]. In this example both trails have a common cycle.

In section 5.5.2 we defined T_u as the set of arcs on the trail from u to the last node on the trail and $\gamma_u = 1$ if u is the last node on the trail. For a nonbasic arc $x_k(i,j)$ for which the capacity is to be varied, this is no longer true. We must adjust the γ's at the origin and the terminal nodes of $x_k(i,j)$ to account for the effects of the parametric change we are going to make on arc $x_k(i,j)$. In particular for the origin node of arc $x_K(i,j)$, node i, which is the last node of one trail, $\gamma_i = p_k$. For the terminal node of arc x_k, node j, which is the last node on the second trail, $\gamma_j = -p_k a_k$.

Each trail generated by a nonbasic arc x_k will create node gains for each node on the trail. The following examples will demonstrate that by using a series of additive operations, γ_u can be computed depending on where node u is located on the trail.

<u>Example 1:</u> Consider the network shown in Figure 5.1(b). As noted earlier, candidate arc 20 forms two flow augmenting trails. Calculating the node gains for the flow augmenting trail starting at node 11 and ending at node 9, we have

$$\gamma_9 = p_{20}$$
$$\gamma_6 = p_{20}/a_{14}$$
$$\gamma_3 = p_{20}/(a_{14}a_6)$$
$$\gamma_1 = p_{20}/(a_{14}a_6a_2)$$
$$\gamma_{11} = p_{20}/(a_{14}a_6a_2a_{22}).$$

However we must also consider the second flow augmenting trail from node 11 to node 10. In this case we calculate the node gains as

$$\gamma_{10} = -p_{20}a_{20}$$
$$\gamma_7 = -p_{20}a_{20}/a_{15}$$
$$\gamma_5 = -p_{20}a_{20}/(a_{15}a_{11})$$
$$\gamma_2 = -p_{20}a_{20}/(a_{15}a_{11}a_5)$$
$$\gamma_3 = -p_{20}a_{20}/(a_{15}a_{11}a_5a_7)$$

$$\gamma_1 = -p_{20}a_{20}/(a_{15}a_{11}a_5a_7a_2)$$
$$\gamma_{11} = -p_{20}a_{20}/(a_{15}a_{11}a_5a_7a_2a_{22}).$$

We note that nodes 3, 1, and 11 have been encountered twice, once with the flow augmenting trail ending at node 9 and once with the flow augmenting trail ending at node 10. Thus the node gains for nodes 3, 1, and 11 must be adjusted for these nodes on a common augmenting trail. Therefore

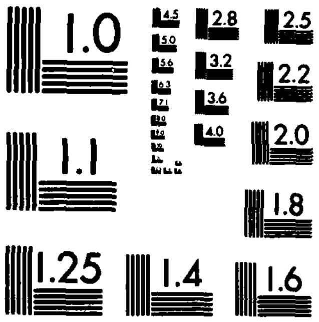
$$\gamma_3 = p_{20}/(a_{14}a_6) - p_{20}a_{20}/(a_{15}a_{11}a_5a_7)$$
$$\gamma_1 = p_{20}/(a_{14}a_6a_2) - p_{20}a_{20}/(a_{15}a_{11}a_5a_7a_2)$$
$$\gamma_{11} = p_{20}/(a_{14}a_6a_2a_{22}) - p_{20}a_{20}/(a_{15}a_{11}a_5a_2a_{22}).$$

Since the network is linear, we account for the node gains on the common part of the flow augmenting trail by adding the node gain values of the first flow augmenting trail to the node gains of the second flow augmenting trail. The node gains on the unique part of the flow augmenting trails remain unchanged.

Example 2: Consider the network shown in Figure 5.2. We will now calculate the node gains for the flow augmenting trails generated by arc 15. To calculate the node gains for the flow augmenting trail starting at node 6 and terminating at node 7 we have :

AD-A145 558 PARAMETRIC ANALYSIS FOR GENERALIZED NETWORK FLOW PROBLEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH M E BAUM MAY 84 AFIT/CI/NR-84-38D 2/3

UNCLASSIFIED F/G 12/1 NL


1.0
4.5
2.8
2.5
5.0
5.6
3.2
2.2
6.3
3.6
7.1
4.0
2.0
1.1
1.8
1.25
1.4
1.6

$$\gamma_7 = p_{15}$$
$$\gamma_5 = [p_{15}/a_{11}]\ (\beta/\beta - 1)$$
$$\gamma_2 = [p_{15}/(a_{11}a_5)]\ (\beta/\beta - 1)$$
$$\gamma_3 = [p_{15}/(a_{11}a_5a_7)]\ (\beta/\beta - 1)$$
$$\gamma_6 = [p_{15}/(a_{11}a_5a_7a_{-6})]\ (\beta/\beta - 1).$$

To calculate the node gains for the flow augmenting trail starting at node 3 and terminating at node 10, we have :

$$\gamma_{10} = -p_{15}a_{15}$$
$$\gamma_9 = -p_{15}a_{15}/a_{20}$$
$$\gamma_4 = -p_{15}a_{15}/(a_{20}a_{-19})$$
$$\gamma_6 = [-p_{15}a_{15}/(a_{20}a_{-19}a_{-9})]\ (\beta/\beta - 1)$$
$$\gamma_5 = [-p_{15}a_{15}/(a_{20}a_{-19}a_{-9}a_{12})]\ (\beta/\beta - 1)$$
$$\gamma_2 = [-p_{15}a_{15}/(a_{20}a_{-19}a_{-9}a_{12}a_5)]\ (\beta/\beta - 1)$$
$$\gamma_3 = [-p_{15}a_{15}/(a_{20}a_{-19}a_{-9}a_{12}a_5a_7)]\ (\beta/\beta - 1).$$

We note that nodes 2, 3, 5, 6 appear twice. Thus these node gains must be adjusted since they apppear on a common augmenting trail. For example the adjusted node gain of node 2 is

$$\gamma_2 = [p_{15}/(a_{11}a_5) - p_{15}a_{15}/(a_{20}a_{-19}a_{-9}a_{12}a_5)]\ (\beta / \beta - 1)$$

and the node gains for nodes 3, 5, and 6 may be adjusted similarly. The node gains on the unique parts of the flow augmenting trails remain unchanged.

5.5.3 Capacity Change on a Set of Nonbasic Arcs

When we consider more than one nonbasic arc, we must take into account the combined effects of numerous intersecting flow augmenting trails. We should note that if any of the nonbasic arcs have zero flow, no further analysis is needed for this arc. The arc will remain nonbasic as the change on the capacity will have no effect on the flow since the flow is at zero.

If we consider the example network in Figrure 5.2, we see that there are two nonbasic arcs, arcs 14 and 15. For each candidate arc, two flow augmenting trails are produced. For arc 15, one trail starts at a cycle and ends at node 7 with arcs [11,5,7,-6,12] and nodes [7,5,2,3,6]. The second trail created by arc 15 has arcs [20,-19,-9,12,5,7,-6] and nodes [10,9,4,6,5,2,3]. For arc 14 the first flow augmenting trail has arcs [12,5,7,-6] and nodes [6,5,2,3]. The second trail created by arc 14 has arcs [-19,-9,12,5,7,-6] and nodes [9,4,6,5,2,3].

In each case to calculate the γ values for each node on the trails, we see that certain nodes are on unique trails, while other nodes are part of intersecting trails. For example, node 7 has a unique γ value as no other flow augmenting trails pass through this node. In particular

$$\gamma_7 = p_{15}.$$

This is contrasted to node 6 which is on flow augmenting trails which end at nodes 7, 10, and 9 respectively. Thus γ_6 must take into account the additive contributions from each of the trails which pass through node 6. In particular

$$\gamma_6 = [p_{14} - p_{14}a_{14}/(a_{-19}a_{-9}) + p_{15}/(a_{11}a_5a_7a_{-6}) \\ -p_{15}a_{15}/(a_{20}a_{-19}a_{-9})]\ (\beta / \beta - 1).$$

In order to write the general formulas to calculate the node gains, we need some additional notation. Define T_u to be the set of flow augmenting trails that pass through node u. Let $T_{u,\tau}$ be the set of arcs on the trail $\tau \epsilon T_u$ that originate at node u and go to the end of the trail. Then we can write the general formulas for the calculation of the node gains as

$$\gamma_u = \sum_{\tau \in T_u} \rho_\tau / (\prod_{k \in T_{u,\tau}} a_k) \quad \text{u not on a cycle} \qquad (5.3)$$

$$\gamma_u = (\beta/\beta - 1) \sum_{\tau \in T_u} \rho_\tau / (\prod_{k \in T_{u,\tau}} a_k) \quad \text{u on a cycle} \quad (5.4)$$

$$\text{where } \rho_\tau = \begin{cases} p_k \text{ if the trail ends at the origin node of candidate arc } x_k \\ -p_k a_k \text{ if the trail ends at the terminal node of candidate arc } x_k. \end{cases}$$

5.5.4 Capacity Change in a Set of Candidate Arcs

When we consider a set of candidate arcs in a network problem, each arc must have one of three forms. First, a candidate arc may be basic. If this is the case, increasing the capacity will have no effect on the flows on any other basic arc flows as the candidate arc under consideration would still have its flow between zero and its upper bound and therefore remain basic. Decreasing the capacity of this basic arc will likewise have no effect until the capacity is decreased to equal the flow on the arc. We note at this point a basis change will occur.

The second possibility is that a candidate arc may be nonbasic and have zero flow. If the capacity of the candidate arc is increased, there will be no change in any of the flows in the basic arcs as the candidate arc under consideration will remain nonbasic with zero flow. If the capacity is being reduced no effect will occur. However, we should note that when the candidate arc's capacity becomes equal to zero, the candidate arc is nonbasic at its upper bound and that any further reductions in the candidate arc's capacity will cause the problem to become infeasible.

Lastly, we must consider nonbasic candidate arcs which have flow equal to their upper bound. Changes in the capacity by either increasing or decreasing the nonbasic arc under consideration will cause flow in the arc to change simultaneously. These arcs will remain nonbasic at their upper bounds. However, we must adjust the basic flows to accommodate these flow changes in the nonbasic candidate arcs at their upper bound in the manner of section 5.5.3.

5.5.5 Computer Implementation of Flow Augmenting Trails

The subroutines used to calculate the node gains for the flow augmenting trails are PATH, TRAIL2, and CYCLEG. Subroutine PATH is called by the main program PSENG when a flow augmenting trail is to be calculated for a nonbasic candidate arc with flow equal to its upper bound. TRAIL2 is called by PATH when a node is encountered on a common part of a flow augementing trail. CYCLEG is used to compute the gain of any of the cycle(s) found during PATH and to update the node gain for each of the nodes on the cycle(s).

For the computer implementation of the concepts developed in section 5.5.1 through 5.5.4, we need to define the variables which are used in the BASIC code. For each node on the flow augmenting trail(s) we assign a node check value, denoted IK(IJ), to enable us to determine if a node has been previously encountered on a flow augmenting trail. We use the computer coding technique to set the value of MULT to equal +1 if we are begining the trace of a flow augmenting trail that terminates at the origin node of x_k, or we set MULT equal to -1, if we are starting a trace of a flow augmenting trail that ends at the terminal node of x_k. LN refers to the list of nodes encountered and LA refers to the list of arcs encountered on the trail(s). IC is a counter to indicate the number of nodes we have encountered and CK is an indicator if this node has only been encountered on this trail (ie, on a cycle). The RC(Z) list is a list of root nodes of all the cycles encountered during the various passes through the flow augmenting trail(s). The G(IJ) list contains the node gain values for each node encountered on the flow augmenting trail.

We should note that the subroutine PATH in conjunction with TRAIL2 and CYCLEG differ from the subroutine PATHPG [46, pg. 323] in three ways. First, for each candidate arc x_k, the parametric parameter must be taken into consideration. Second, the γ's are calculated additively for each flow augmenting trail created by a candidate arc. Finally, if a cycle is encountered, the root node of the cycle is entered on the root cycle list [RC(Z)]. Then after all the flow augmenting trails have been generated, the cycle gains are calculated for each cycle in the root cycle list and then each node gain on a particular cycle is updated.

As an example of the use of PATH, consider the application to the network shown in Figure 5.2. The following information is known concerning the arcs whose capacities are being varied. The candidate arcs are

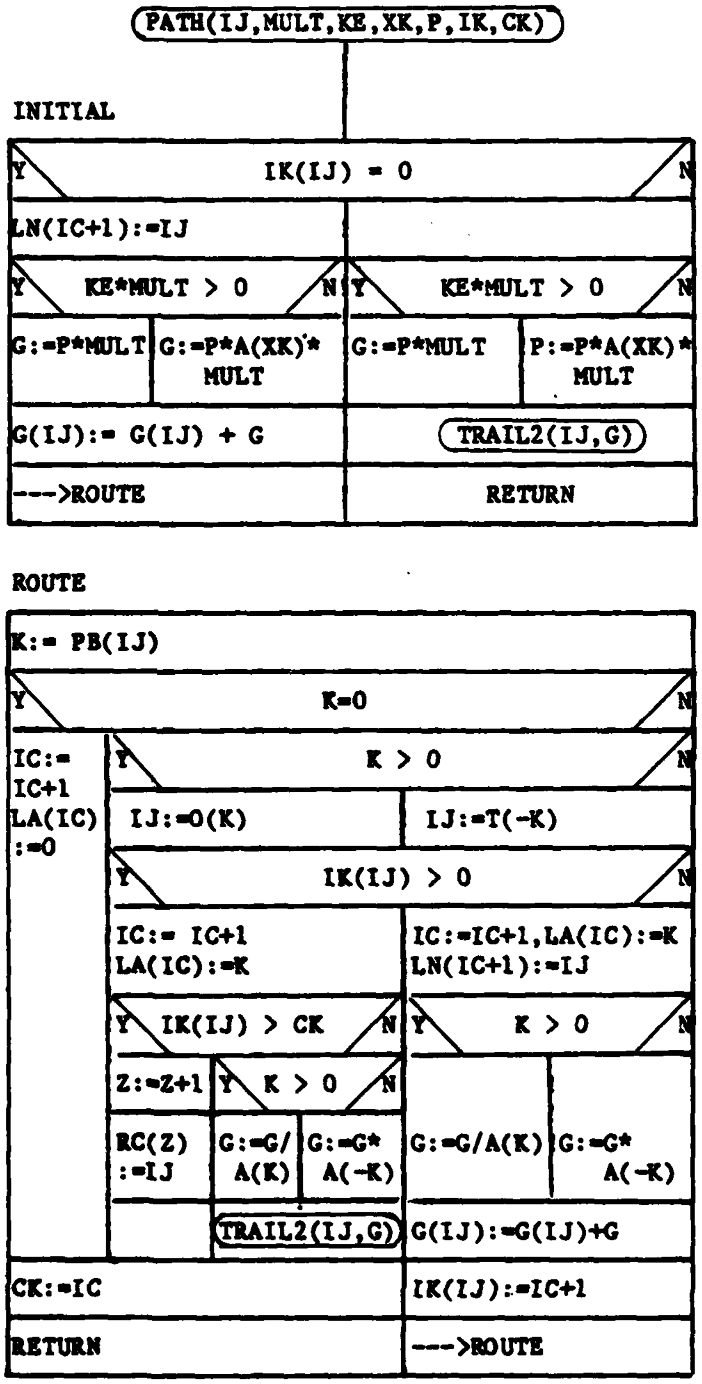
$$X = [5,6,7,11,14,15,20]$$

and the parametric values are

$$P = [-.3,1.2,-2.1,-0.67,0.1,-3,1.4].$$

We note that only arcs 14 and 15 are considered in this analysis as they are the only nonbasic candidate arcs at their upper bounds. The list of basic arcs at the current optimal solution is

PURPOSE: To find the flow augmenting trail for a generalized network and compute γ's for each node on the trail.
1.(INTIAL) If node IJ has never been encountered (IK(IJ)=0),put IJ in LN and calculate γ for node IJ. Otherwise node IJ was visited before. Update node gains with TRAIL2 and RETURN.
2.(ROUTE) Identify arcs on the flow augmenting trail. Find the back pointer of the current node. If zero the slack node has been visited. If not find next node in trail and check if visited. If not add IJ to LN and K to LA. Update node gain and node check value , go to 2. Otherwise,check if node only visit- on this path (IK > CK).If so put IJ on root cycle list and return. Otherwise, IJ has been visited on a previous flow augmenting trail. Call TRAIL2 to update γ's on common trail and return.

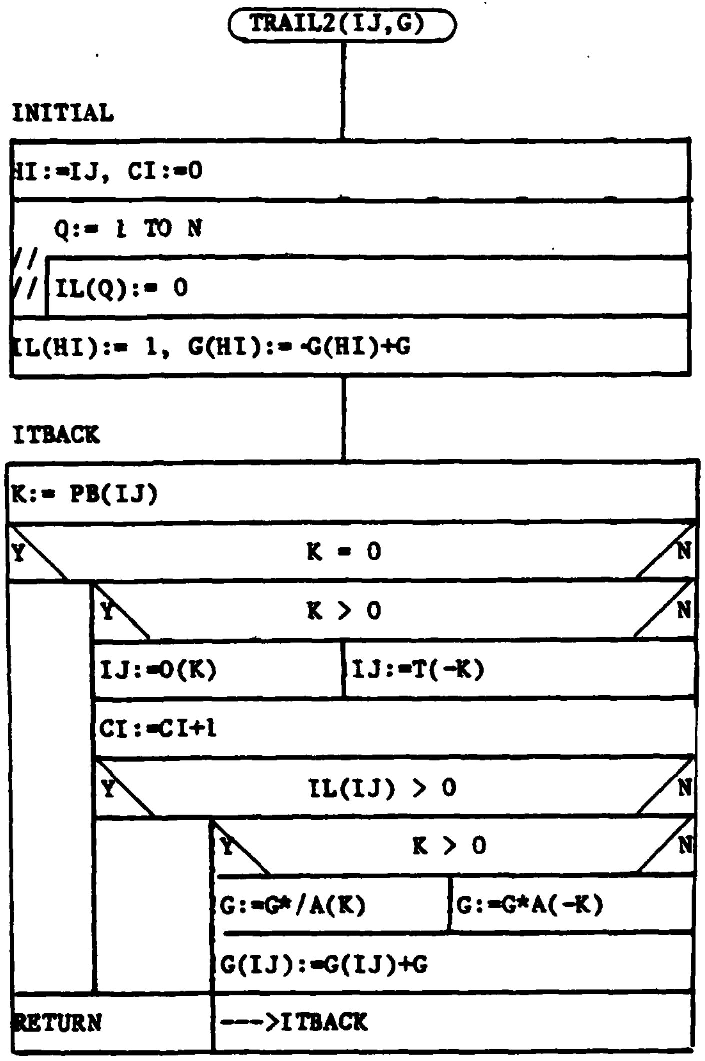


Subroutine PATH
Figure 5.4

PURPOSE: Update values on common trail found in PATH.

1.(INTIAL) Set all node check values to zero. Update γ value of common node IJ.

2.(ITBACK) Look at back pointer of IJ. If it is zero, the slack node has been encountered, return. If not check if node has been visited before, if not update check value and γ for node. Go to 2. If node has been visited before, a root of a cycle has been found, return.

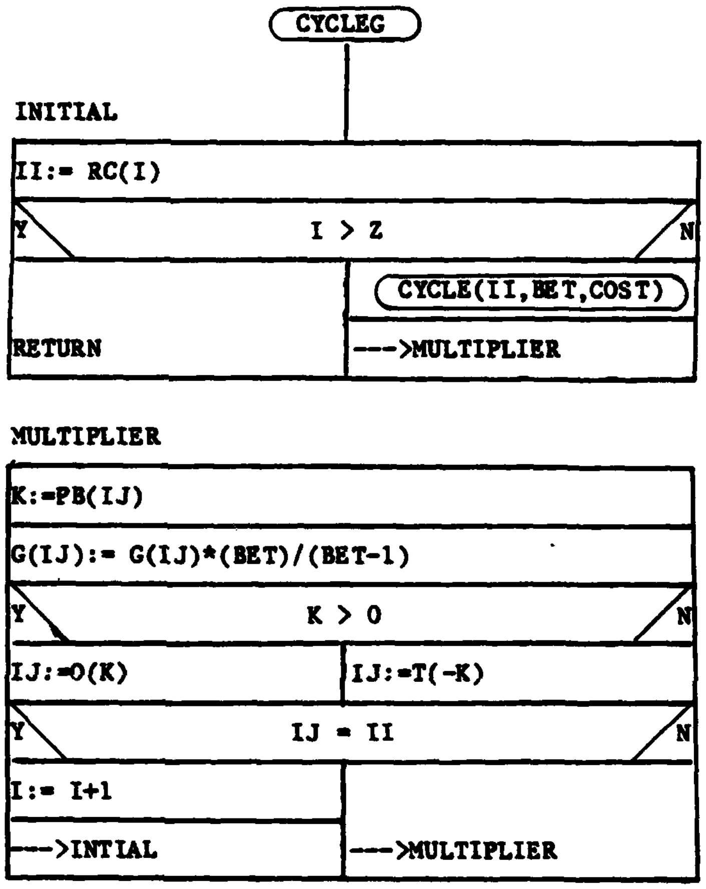


Subroutine TRAIL2
Figure 5.5

PURPOSE: Compute the gain of a cycle defined by the back pointers and to update the node gains for each node on a cycle by its cycle gain.

1.(INITIAL) Set II to be root node of cycle from cycle list. If all cycles have been updated (I > Z), return. Otherwise calculate cycle gain for cycle RC(I) by calling CYCLE.

2.(MULTIPLIER) Update γ values for for each node on cycle RC(I). Update I and go to 1.



Subroutine CYCLEG
Figure 5.6

[-3,5,-6,7,-9,11,12,-18,-19,20].

The basic arcs are shown in Figure 5.2 as solid lines and the candidate arcs at capacity by dashed-dot lines.
At the conclusion of PATH for the candidate arc 14 ending at node 6, the following information has been obtained:

Node List: LN = [6,5,2,3]

Arc List : LA = [12,5,7,-6]

Nodes Visited: IK = [0,3,4,0,2,1,0,0,0,0,0]

Node Gains: G = [0,0.057,0.067,0,0.143,0.1,0,0,0,0,0]

Cycles Encountered: Z = 1

Root Cycle List : RC(1)= 6

Nodes Encountered: CK = 4.

We have simply followed the back pointers from node $i_{14} = 6$ until a node is encountered that has been visited before. In this case the node is node 6, since a cycle is formed. The list IK serves the purpose to mark nodes that have been visited. At this point node 6 is entered on the root cycle list [RC(1) = 6] and CK is set to 4 to indicate a cycle has been found on this augmenting path. LN contains the nodes in the order that they were encountered by PATH, and LA contains the backpointers of the nodes in LN. These backpointers comprise the list of arcs in the trail. IC is a counter to indicate how many nodes have been encountered. The G list indicates the value of γ_u for each node.

At the completion of PATH for arc 14 ending at node 9, the lists have been updated to read:

LN = [6,5,2,3,9,4]

LA = [12,5,7,-6,-19,-9]

IK = [0,3,4,6,2,1,0,0,5,0,0]

G = [0,-.57,-.06,-.25,-.14,-.1,0,0,-.1,0,0]

CK = 6   Z = 1   RC(1) = 6.

Note that when node 6 is encountered IK(6) = 1 and CK = 6 indicating this node had been encountered on a previous cycle and the γ_u's need only be updated for the common part of the flow augmenting path. Recall the effects of the cycle gain, $\beta/(\beta - 1)$, will be accounted for after the calculations for all the flow augmenting paths are completed.

We must now also add in the effects for the candidate arc, $x_k = 15$. After the completion of PATH ending at node 10, the lists have been updated again to be:

LN = [6,5,2,3,9,4,10]

LA = [12,5,7,-6,-19,-9,20]

IK = [0,3,4,6,2,1,0,0,5,7,0]

G = [0,2.85,3.36,6.12,7.14,5,0,0,2.45,2.55,0]

CK = 7 Z = 1 RC(1) = 6.

In ROUTE node 9 is again encountered, so the additive effects for the common augmenting path updates the G list.

After the completion of PATH ending at node 7, the lists have been updated to read:

LN = [6,5,2,3,9,4,10,7]

LA = [12,5,7,-6,-19,-9,20,11]

IK = [0,3,4,6,2,1,8,0,5,7,0]

G = [0,1.65,1.94,6.12,4.14,3.72,-3,0,2.45,2.55,0]

CK = 9 Z = 1 RC(1) = 6.

As there are no more candidate arcs to be considered, the cycle gain for the cycle encountered during PATH needs to be calculated. In this case it is rooted at node 6 and contains the nodes 2, 3, 5, and 6. The cycle gain is 1.65 and the updated G list looks like:

$$G = [0,4.19,4.93,6.12,10.49,9.44,-3,0,2.45,2.55,0]$$

and the lists LN, LA, and IK remain unchanged.

5.6 Computing Maximum Parametric Variation of Capacities and Finding the Arc to Leave the Basis

As in the previous sections, we assume we have an optimal solution to a given network flow problem. We define the maximum reference number, r, as the maximum flow augmentation that will drive a basic flow to zero or capacity or drive a candidate arc's upper bound to zero. Then as we change the capacity on a candidate arc x_k, the new capacity on the candidate arc will be

$$c'_k = c_k + p_k r \qquad (5.5)$$

where $k \in X$ and c_k is the capacity, p_k is the parametric parameter, and r is the reference number of the candidate arc.

We now need to determine the maximum positive increment r_k that will drive a basic flow to a bound or cause the capacity on a candidate arc to go to zero. Assume there are initial flows in the arcs on the flow augmentation trail(s), identified as f_k or f_{-k} depending on whether k is a forward or a mirror arc. We should note that since arc k is basic and if p_k is not equal to zero, the capacity on the basic arc is changing as well as the flow. Thus we must monitor the simultaneous effects of both changes. Recalling that r_k is defined as the maximum flow change that will drive the flow in arc k to one of its bounds, we need to consider two cases.

CASE 1: k is a forward arc

If $k > 0$, then the flow change in the original network implied by a flow increment r is

$$\Delta_k = \gamma_j r / a_k$$

where γ_j is the node gain at the terminal node of arc k. Since the initial flow on arc k is defined by f_k, then the flow in arc k after a flow increment is

$$f_k + \Delta_k = \Delta_k + \gamma_j r / a_k.$$

Since r is greater than or equal to zero, the direction of the flow change depends only on the sign of γ_j / a_k.

SUBCASE A $\gamma_j / a_k \geq 0$: With γ_j / a_k positive, the arc flow increases. Since r is the value of the reference number that will drive a basic flow to zero or a new capacity, $c_k + p_k r$, the new flow will be limited by

$$f_k + (\gamma_j r/a_k) \leq c_k + p_k r$$

or the value of r that drives the basic arc k to its upper bound is

$$r_k = (c_k - f_k)a_k/(\gamma_j - p_k a_k). \qquad (5.6)$$

We should note that if $\gamma_j - p_k a_k \leq 0$, the new capacity is increasing quicker than the flow is changing. This means this arc would remain basic since the new flow will always remain less than its new capacity. We set $r_k = \infty$, since it will not create a limiting value.

SUBCASE B $\gamma_j/a_k < 0$: In this case, since the factor γ_j/a_k is negative, the flow in arc k is decreasing with r. The new flow will be limited by

$$0 \leq f_k + \gamma_j r/a_k \leq c_k + p_k r.$$

However, two possibilities can occur. If $\gamma_j - p_k a_k \leq 0$, the flow is going to zero quicker than the capacity and therefore the limiting value of r for arc k is

$$r_k = - f_k a_k/\gamma_j \qquad (5.7)$$

or the capacity could be decreasing faster than the flow is decreasing, $(\gamma_j - p_k a_k) > 0$, so in this case

$$r_k = (c_k - f_k)a_k/(\gamma_j - p_k a_k). \qquad (5.8)$$

CASE 2: k is a mirror arc

If $k < 0$, then the flow change in the original network implied by a flow increment r is

$$\Delta_{-k} = -\gamma_j r$$

where $j = o(-k)$. Since the initial flow on arc k is defined by f_{-k}, then the flow on arc k after a flow change is

$$f_{-k} + \Delta_{-k} = f_{-k} - \gamma_j r.$$

SUBCASE A $\gamma_j < 0$: With $\gamma_j < 0$, the arc flow will increase and the value of r which drives it to the new capacity, $c_{-k} + p_{-k}r$, will be limited by

$$f_{-k} - \gamma_j r \leq c_{-k} + p_{-k} r$$

or the limiting value when $\gamma_j + p_{-k} < 0$ is

$$r_k = (f_{-k} - c_{-k})/(\gamma_j + p_{-k}). \tag{5.9}$$

If $\gamma_j + p_{-k} \geq 0$ the new flow is increasing at a slower rate than the capacity. This means this arc would remain basic since the new flow will be less than the new capacity. Therefore we set $r_k = \infty$, as it will not create limiting value.

SUBCASE B $\gamma_j > 0$: In this case, since the factor $\gamma_j > 0$, $-\gamma_j r$ is negative, and the flow in arc k is decreasing. Thus the flow will be limited by

$$0 \leq f_{-k} - \gamma_j r \leq c_{-k} + p_{-k} r.$$

Again two cases exist. If $\gamma_j + p_{-k} \geq 0$, the flow is going to zero faster than the capacity and thus

$$r_k = f_{-k}/ \gamma_j. \tag{5.10}$$

Otherwise the capacity could be decreasing faster than the flow is decreasing, so

$$r_k = (f_{-k} - c_{-k})/(\gamma_j + p_{-k}). \tag{5.11}$$

For each of the six possibilities listed above in equations (5.6) through (5.11), r_k is positive. The maximum reference number or flow increment for the trail(s) is the reference number that will drive the flow on one or more arcs to zero or its capacity without causing any other flow to be infeasible. Thus

$$r_1 = \min_{k \varepsilon M_Q} \{r_k\} \qquad (5.12)$$

where M_Q is the set of arcs on the flow augmenting trail(s) and r_k is determined as in equations (5.6) through (5.11).

We must also consider the possiblity that a nonbasic candidate arc will have its capacity to zero. Thus for each nonbasic candidate arc with $p_k < 0$, we must calculate

$$r_k = c_k/-p_k. \qquad (5.13)$$

If a nonbasic candidate arc has a positive parametric parameter, $p_k > 0$, we set $r_k = \infty$ as this arc will not create a limiting value of r.

Again we need to calculate the maximum reference number for the nonbasic candidate arcs

$$r_2 = \min_{k \in X_R} \{r_k\} \tag{5.14}$$

where X_R is the set of of nonbasic candidate arcs and r_k is determined as in equation (5.13).

We must pick the smallest of the reference numbers calculated in (5.12) and (5.14), as this will be the maximum reference number or flow increment r that will either drive a basic arc to zero or capacity or a nonbasic arc's capacity to zero. Thus

$$r = \text{Min}\ \{r_1,\ r_2\}\ .$$

The MFLOG2 subroutine is use to find the maximum flow change in the trail, given the γ values and the list of arcs and nodes in the trail(s). PSENG calls MFLOG2 after all the flow augmenting trail(s) have been scanned. MFLOG2 will determine the basic arc that limits the flow change on the flow augmenting trail(s) and thus the arc which may leave the basis. As in Chapter 10 (46), the FLOWG subroutine modifies the arc flows for a given trail or trail(s) and a given value of r.

To illustrate some of the types of calculations that are performed in MFLOG2, consider the optimal basis shown in Figure 5.2. The list of candidate arcs are

[5,-6,7,11,14,15,20]

and the associated parametric parameters are

[-0.3,1.2,-2.1,-0.67,0.1,-3,1.4].

The basic arcs on the flow augmenting trails are

[12,5,7,-6,-19,-9,20,11].

At the end of PATH we have calculated the following values for then nodes 1 through 11

G = [0,4.19,4.93,6.12,10.49,9.44,-3,0,2.45,2.55,0].

As an example of the calculations preformed in MFLOG2 consider arc 20. For arc 20 we have $\gamma_{10}/a_{20} = 2.55$ and this means we are in case 1. Calculating r_{20}, we have

$$\begin{aligned} r_{20} &= (c_{20}-f_{20})a_{20}/(\gamma_{10} - p_{20}a_{20}) \\ &= .843. \end{aligned}$$

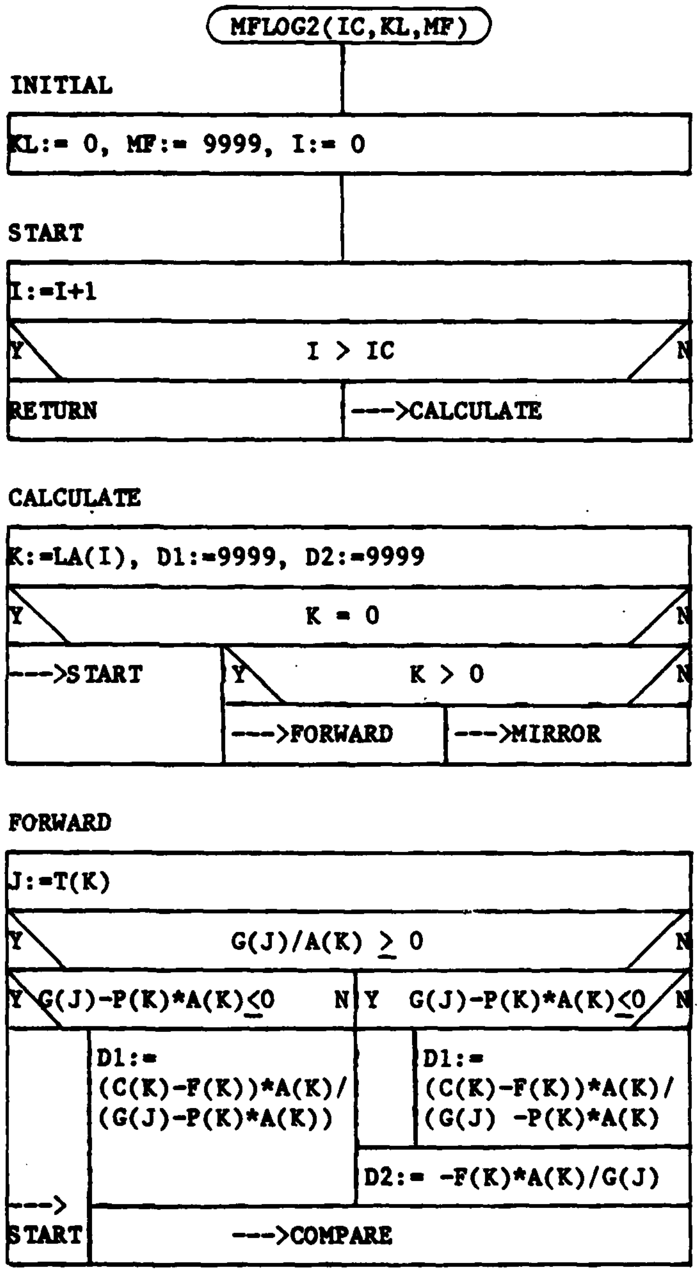
PURPOSE: To find the maximum flow augmentation possible on all basic arcs in LA created by flow augmenting trails.

1.(INITIAL) Set the leaving arc to zero, the maximum flow to a large number, and loop counter to zero.

2.(START) Increment the loop counter. Check if all arcs in LA have been checked. If so return. Otherwise go to 3.

3.(CALCULATE) Set k to the LA(I). If k is zero, go to 2. Otherwise check if k is a forward arc. If so go to 4, otherwise go to 5.

4.(FORWARD) Set j equal to the terminal node of k. Determine the appropriate values of D1 and D2. Go to 6.

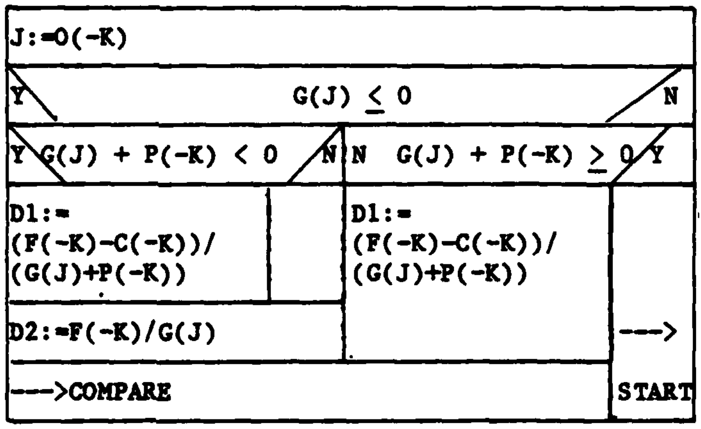


Subroutine MFLOG2
Figure 5.7

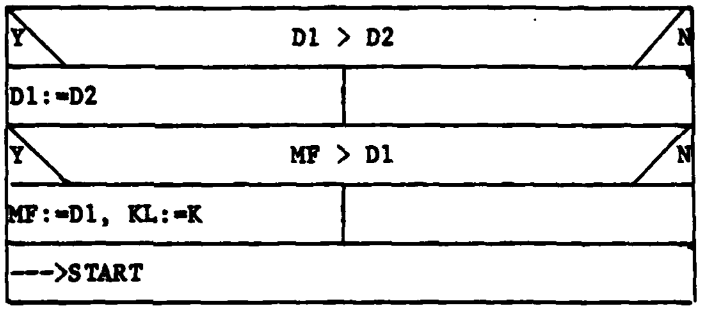
5.(MIRROR) Set j to the origin of node -k. Calculate D1 and D2 as appropriate. Go to 6.

6.(COMPARE) If the new value of D1 or D2 is less than MF, replace MF with the new value and keep track of the leaving arc. Go to 2.

MIRROR



COMPARE



Subroutine MFLOG2
Figure 5.7(cont)

If we consider arc 11, we are in SUBCASE A of case 1 where $k > 0$ and $\gamma_j < 0$. We note that $\gamma_7 - p_{11}a_{11} \leq 0$ and therefore

$$\begin{aligned} r_{11} &= -f_{11}a_{11}/\gamma_7 \\ &= 1.2. \end{aligned}$$

We may determine the remaining reference numbers for the basic arcs by the methods discussed in this section and these values are:

$$\begin{aligned} r_5 &= 3.941 \\ r_{-6} &= 0.9529 \\ r_7 &= 4.608 \\ r_{-9} &= 0.8257 \\ r_{12} &= 0.1482 \\ r_{-19} &= 0.5482 \end{aligned}$$

The minimum occurs for arc 12. Therefore the maximum reference number for the basic arcs is

$$r = 0.1482$$

5.7 Determining the Entering Arc

In the previous sections we developed the mechanisms to find an arc to leave the basis. Since this is a dual algorithm, we now need to develop the techniques to be able to select an admissible entering arc.

5.7.1 Characteristics of the Entering Arc

An entering arc, k_E, must have two characteristics to enable it to enter the network. The first characteristic is that the entering arc must be admissible. Since the algorithm is applied to the marginal network representation that admits both forward ($k > 0$) and mirror ($k < 0$) arcs, we define the following admissibility rule:

$k > 0$ and $c_k - f_k > 0$ then arc k is admissible

$k < 0$ and $f_{-k} > 0$ then arc $-k$ is admissible.

We can state this in a more intuitive manner as

a forward arc is admissible if the flow in the original arc is less than its capacity and a mirror arc is admissible if the flow in the original arc is greater than zero.

The second characteristic of the entering arc concerns its location in the network relative to the leaving arc. When the leaving arc is deleted from the basis, it divides the nodes of the network into sets N_1 and N_2. The set of nodes which remains connected to the terminal node of the leaving arc k_L (j_L) is designated the N_2 set. We will denote the basis subnetwork containing N_2 as D_2. We note that D_2 always forms a tree rooted at the terminal node of the leaving arc and no longer includes the slack node or a cycle. The set N_1 is the complementary set of nodes, that is $N_1 = N - N_2$. We denote the basis subnetwork that contains N_1 as D_1. Thus D_1 consists of one or more components that always includes the slack node and any flow-generating cycles that remain after we delete the leaving arc from the basis.

We may mathematically express the components that are formed when $k_L(i_L,j_L)$ is removed from the optimal basis tree, $D = [N,M_T]$ as

$$D_1 = [N_1,M_1]$$
$$D_2 = [N_2,M_2]$$

where

$i_L \varepsilon N_1 \qquad j_L \varepsilon N_2$

and

$$N_1 \cup N_2 = N$$

$$M_1 \cup M_2 = M_T - k_L.$$

When arc k_L, which may be a forward arc ($k_L > 0$) or a mirror arc ($k_L < 0$) depending on the orientation required for the spanning tree, is deleted from the basis, there no longer exists a way to continue augmenting flow on the candidate arcs. Thus the problem is to find an admissible arc to add to the basis network that will reestablish a flow augmentation path to node j_L and therefore allow continued flow augmentation on the candidate arcs. If we let $k \varepsilon M_A$, where M_A is the set of admissible arcs, k may enter the basis in one of three ways. First k may join N_1 to N_2 which we may write as $k \varepsilon (N_1,N_2)$. The notation (N_1,N_2) defines the set of arcs each originating at a node in the set N_1 and terminating at node in N_2. Secondly, k may join N_2 to N_1, which we write as $k \varepsilon (N_2,N_1)$. Lastly k could join N_2 to N_2, which we write as $k \varepsilon (N_2,N_2)$. In the first two cases the subnetwork D_2 is joined to a tree rooted at the slack node or at a cycle in D_1. In the later case a new component of the basis network is formed from D_2 and k_E. This component is rooted at a new cycle.

We now can describe the second characteristic that determines the selection of the entering arc. It must be one of the following four cases, which is illustrated by Figure 5.8.

CASE 1: k_L is a forward arc and leaves the basis at its upper bound

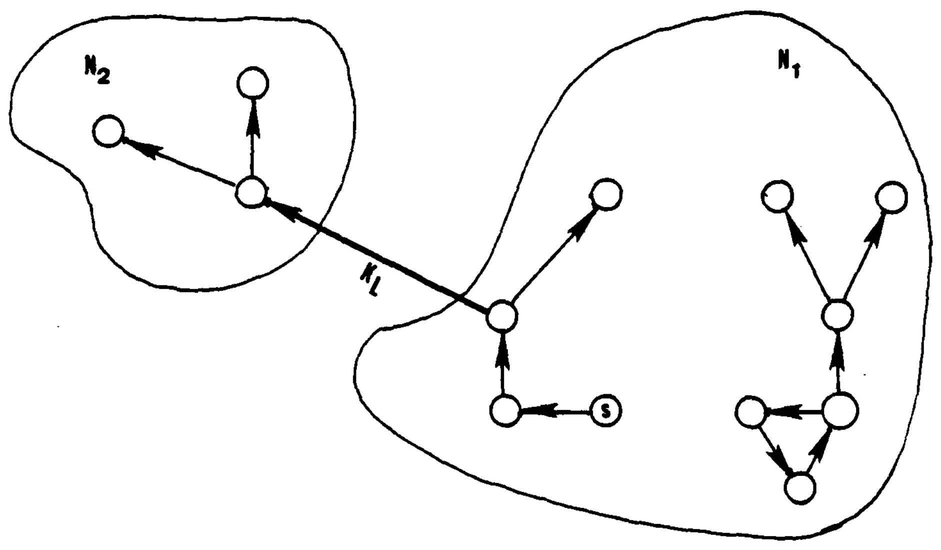
As k_L goes to its upper bound, we may think of additional flow being put into N_2. Thus as k_L leaves the basis, k must form a new flow augmenting path to continue to allow flow to be put into the set N_2. Thus we must have $k \varepsilon (N_1,N_2)$ or $k \varepsilon (N_2,N_2)$. When $k \varepsilon (N_2,N_2)$, k forms a flow generating cycle ($\beta > 1$, where β is the cycle gain).

CASE 2: k_L is a mirror arc and leaves the basis at its upper bound

As k_L leaves the basis by going to its upper bound, we may think of flow being put into the set N_1. Thus we need a $k \varepsilon (N_2,N_1)$ or $k \varepsilon (N_2,N_2)$, where $k \varepsilon (N_2,N_2)$ forms a flow absorbing cycle ($\beta < 1$, where β is the cycle gain).

CASE 3: k_L is a forward arc and leaves the basis at its lower bound

As k_L leaves the basis at its lower bound, we are conceptually putting flow into the set N_1. Thus we must have an arc $k \varepsilon (N_2,N_1)$ or $k \varepsilon (N_2,N_2)$, where in the later case k forms a flow



Determining the Entering Arc
Figure 5.8

absorbing cycle. ($\beta < 1$, where β is the cycle gain).

CASE 4: k_L is a mirror arc and leaves the basis at its upper bound

We can again think of flow being transferred into the set N_2 as k_L leaves the basis. Here we must have $k \varepsilon (N_1,N_2)$ or $k \varepsilon (N_2,N_2)$, and in the later case k forms a flow generating cycle ($\beta > 1$, where β is the cycle gain). This will allow a new flow augmentation path to put new flow into the N_2 set.

Cases one and four are identical as to the form of $k \varepsilon M_A$, and cases two and three are the same. Table 5.1 summarizes these results.

5.7.2 On Determining the New Value of γ

When we are trying to determine the entering arc, we must again use the concept of node gain. However, in this situation we need to define node gain differently than for the case of the leaving arc. In this context, γ_u is only defined for the nodes in the set N_2. In this case γ_u is defined to be the product of the arc gains starting at the terminal node of k_L (j_L) to node u, where we define $\gamma_u = 1$ for $u = j_L$. Thus we see that if one unit of flow arrives at j_L and travels through D_2 to node u the flow arriving at

	$K_L > 0$	$K_L < 0$
K_L going to upperbound	k must go from a) N_1 to N_2 or N_2 to $N_2 (\beta > 1)$	k must go from a) N_2 to N_1 or N_2 to N_2 $(\beta < 1)$
K_L going to lower bound	k must go from a) N_2 to N_1 or N_2 to $N_2 (\beta < 1)$	k must go from a) N_1 to N_2 or N_2 to N_2 $(\beta > 1)$

Determining the Entering Arc
Table 5.1

node u is γ_u. The node gains can be computed from the structure of the basis tree and the value of γ_{jL}. For any arc k(i,j) that is included in the tree D_2, we have

$$\begin{aligned} \gamma_j &= \gamma_i a_k \quad \text{for } k \in M_2 \\ i &= o(k) \\ j &= t(k). \end{aligned} \tag{5.15}$$

Node j_L is the root of a tree in N_2. Assume we have listed the arcs of M_2 in predecessor order (that is an arc appears in the list after all its predecessors). Since we are given the value of $\gamma_{jL} = 1$, progressing through the ordered list M_2 we can use equation (5.15) to compute the node gains for each member of N_2. ROOTG [46,page 247] obtains the ordered list M_2 and NEWGAM, as part of FIND2G, calculates the γ values.

We note that if $k \in (N_2, N_2)$, arc k will always form a cycle with gain of

$$\beta = a_k \gamma_j / \gamma_i. \tag{5.16}$$

This equation will be useful in the algorithms described because it allows the calculation of the cycle gain without an explict identification of the arcs on a cycle.

As an example consider Figure 5.1(b) in which arc 2(1,3) leaves the basis. Here $j_L = 3$ and the value of the node gains in N_2 are

$$\gamma_3 = 1$$
$$\gamma_2 = \gamma_3 a_7 = a_7$$
$$\gamma_5 = \gamma_2 a_5 = a_7 a_5$$
$$\gamma_7 = \gamma_5 a_{11} = a_7 a_5 a_{11}$$
$$\gamma_{10} = \gamma_7 a_5 = a_7 a_5 a_{11} a_{15}$$
$$\gamma_8 = \gamma_5 a_{-17} = a_7 a_5 / a_{17}$$
$$\gamma_6 = \gamma_3 a_6 = a_6$$
$$\gamma_4 = \gamma_6 a_{-9} = a_6 / a_9$$
$$\gamma_9 = \gamma_9 a_{14} = a_6 a_{14}.$$

5.7.3 Selecting the Entering Arc

In order to maintain the optimality of the basis, we must select an arc k to enter the basis which satisfies the conditions of 5.7.1 and has the smallest marginal change in cost per unit flow change in the arc k. In order to do this, we must develop the marginal cost equations. We will first develop the equations which deal with $k \varepsilon (N_1,N_2)$ or $k \varepsilon (N_2,N_1)$. After developing these formulas, we will derive the formulas to calculate the marginal cost for k $\varepsilon (N_2,N_2)$.

CASE 1: k_L is a forward arc and is going to an upper bound or k_L is a mirror arc and is going to a lower bound and $k\varepsilon M_A$ is a forward arc.

If we let d_k be the change in cost per unit flow change in the arc k(i,j) where $k\,\varepsilon (N_1,N_2)$, as illustrated in Figure 5.9, we can calculate d_k as

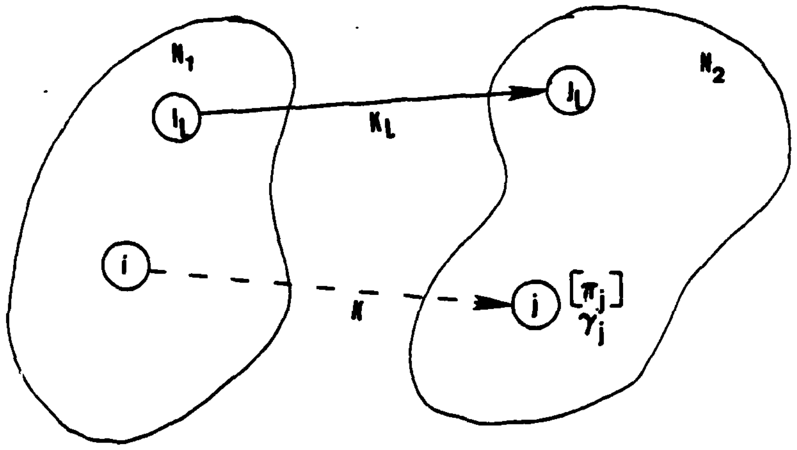
$$d_k = [(\pi_i + h_k)/a_k - \pi_j]\,\gamma_j. \qquad (5.17)$$

This formula is used in the subroutines ABSORB and GENER2.

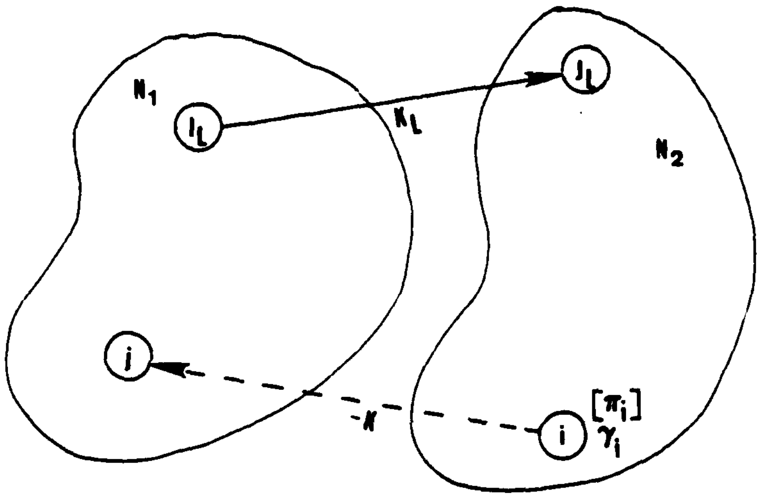
CASE 2: k_L is a forward arc and is going to an upper bound or k_L is a mirror arc and is going to a lower bound and $k\varepsilon M_A$ is a mirror arc.

For arc $k\varepsilon M_A$ and $k < 0$ where $k\,\varepsilon\,(N_2,N_1)$, as illustrated in Figure 5.10, we obtain

$$d_{-k} = [(\pi_j + h_{-k})/a_{-k} - \pi_i]\,\gamma_i.$$



Case 1: Determining the Marginal Arc Cost
Figure 5.9



Case 2: Determining the Marginal Arc Cost
Figure 5.10

Recalling $h_{-k} = -h_k/a_k$ and $a_{-k} = 1/a_k$ in an expanded network,

$$d_{-k} = (\pi_j a_k - h_k - \pi_i)\gamma_i. \qquad (5.18)$$

Equation (5.18) is used in the subroutines ABSORB and GENER2.

CASE 3: k_L is a forward arc and is going to a lower bound or k_L is a mirror arc and is going to an upper bound and $k \varepsilon M_A$ is a forward arc.

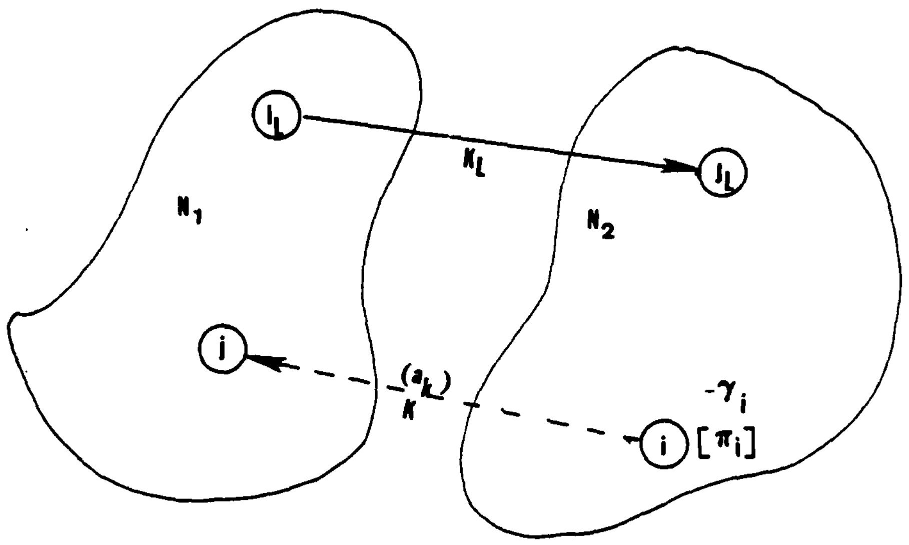
The marginal cost for $k \varepsilon (N_2, N_1)$ per unit flow change in arc k is

$$\begin{aligned} d_k &= [(\pi_i + h_k - a_k \pi_j] [-\gamma_i a_k] \\ &= (\pi_j a_k - h_k - \pi_i)\gamma_i \end{aligned}$$

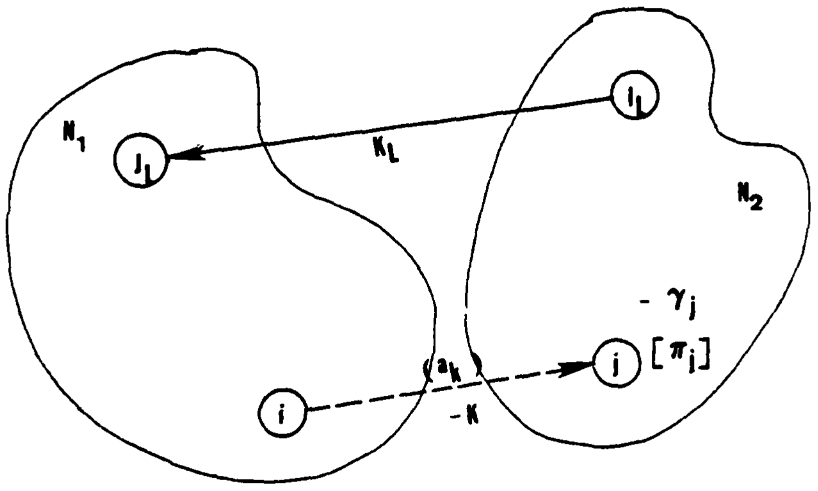
which is identical to equation (5.18). This is illustrated in Figure 5.11.

CASE 4: k_L is a forward arc and is going to a lower bound or k_L is a mirror arc and is going to an upper bound and $k \varepsilon M_A$ is a mirror arc.

The formula for the marginal cost for $k \varepsilon (N_1, N_2)$, $k < 0$, as ilustrated in Figure 5.12, is



Case 3: Determining the Marginal Arc Cost
Figure 5.11



Case 4: Determining the Marginal Arc Cost
Figure 5.12

$$d_{-k} = [(\pi_j + h_{-k} - a_{-k}\pi_i)/a_{-k}]\ [-\gamma_j a_{-k}]$$
$$= [(\pi_i + h_k)/a_k - \pi_j]\ \gamma_j$$

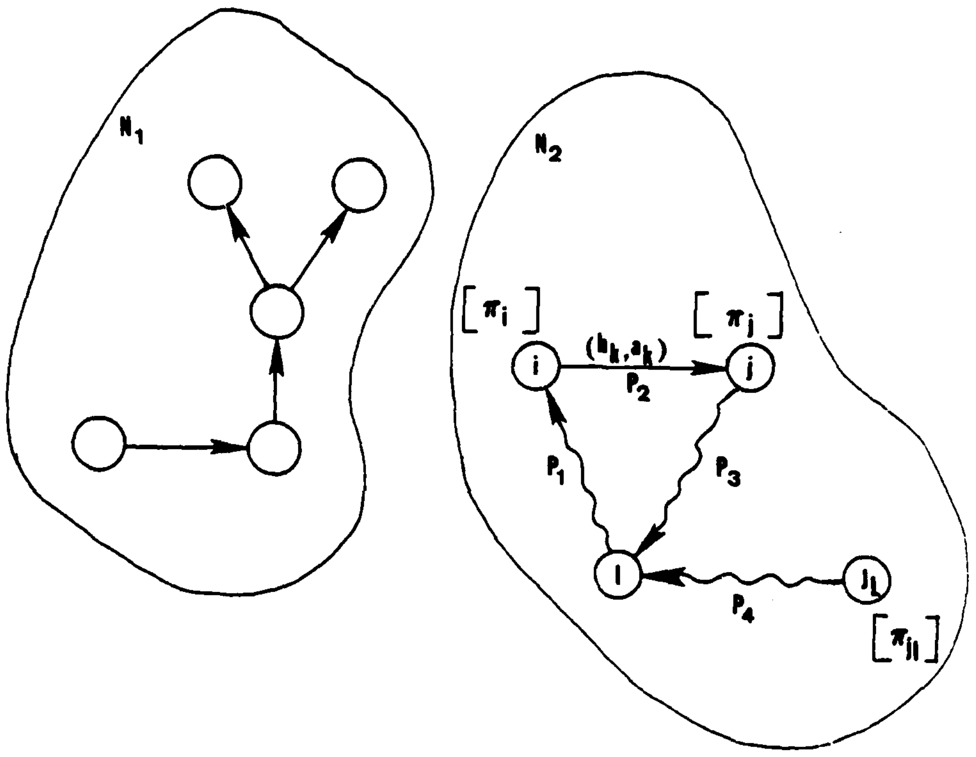
which is the same as equation (5.17).

CASE 5: Arc $k \varepsilon M_A$ and $k \varepsilon (N_2,N_2)$.

In this case an admissible arc k(i,j) where $k\varepsilon (N_2,N_2)$ forms a cycle as shown in Figure 5.13. We must insure the new cycle has a gain different than one. Next we must evaluate the marginal cost of the path formed with the addition of arc k. The path forms four parts.

P_1 : the path from some junction node 1 to i
P_2 : the added arc k
P_3 : the path from j to the junction node 1
P_4 : the path from j_L to 1.

If one unit of flow starts at node j_L, this will require a flow of γ_1 at node 1. The flow into node 1 from the circuit P_3 must be $[\beta / (\beta - 1)]\gamma_1$ and the flow entering P_1 from P_4 and P_3 at node 1 must be $\gamma_1 / (\beta - 1)$, by conservation of flow arguments. The flow out of node j must be $[\beta / (\beta - 1)]\ \gamma_j$ and the flow out node i is $[\beta / (\beta$



Case 5: The Entering Arc Forms a Cycle
Figure 5.13

$- 1)] * [\gamma_j / a_k]$. The cost of the four parts is:

$$P_4 = \pi_{jL} - \pi_1 \gamma_1$$

$$P_3 = [\beta / (\beta - 1)] \; [\pi_1 \gamma_1 - \pi_j \gamma_j]$$

$$P_2 = [\beta / (\beta - 1)] \; [(h_k \gamma_j) / a_k]$$

$$P_1 = [\beta / (\beta - 1)] \; [\pi_i - (a_k \pi_1 \gamma_1) / \beta \gamma_j] \; \gamma_j / a_k.$$

Adding these four parts and simplifying allows us to write

$$d_k = [\beta / (\beta - 1)] \; [(\pi_i + h_k)/a_k - \pi_j] * \gamma_j \qquad (5.19)$$

where $\beta = (a_k \gamma_i)/\gamma_j$ for $k \varepsilon (N_2, N_2)$. This is the formula used in ABSORB and GENER2.

For admissible mirror arcs, $k(j,i)$ $k < 0$, the value of d_{-k} becomes

$$d_{-k} = [\beta / (\beta - 1)] \; [\pi_j a_k - h_k - \pi_i] \gamma_i \qquad (5.20)$$

where $\beta = \gamma_j / (a_k \gamma_i)$ for $-k \varepsilon (N_2, N_2)$. This is the formula used in ABSORB and GENER2.

Recall that in section 5.7.1 we outlined four cases which determined the characteristics of the entering arc. In cases 1 and 4 we needed a flow generating cycle. Using equation (5.16) we may determine if arc k has $\beta > 1$, $\beta < 1$, or $\beta = 1$. If $\beta = 1$, arc k is not admissible. If arc k has $\beta < 1$, and the mirror arc k, $k < 0$, is admissible, we will calculate the marginal cost for arc k. If arc k has $\beta > 1$ we calculate the marginal cost for arc k, $k > 0$. The similar procedure is used for cases 2 and 3 where the entering arc needs to form a flow absorbing cycle.

Since d_k represents the marginal increase in cost per unit flow change in an admissible arc, we need a dual type iteration to find the arc with the smallest value of d_k. This will be the entering arc and receive the designation k_E. Recalling if M_A is the set of admissible arcs from the expanded network such that

if $k(i,j) \varepsilon M_A$ either

$k \varepsilon (N_1,N_2)$, $k \varepsilon (N_2,N_1)$, or $k \varepsilon (N_2,N_2)$.

Then the entering arc is determined by

PURPOSE: To find and arc to enter the basis.

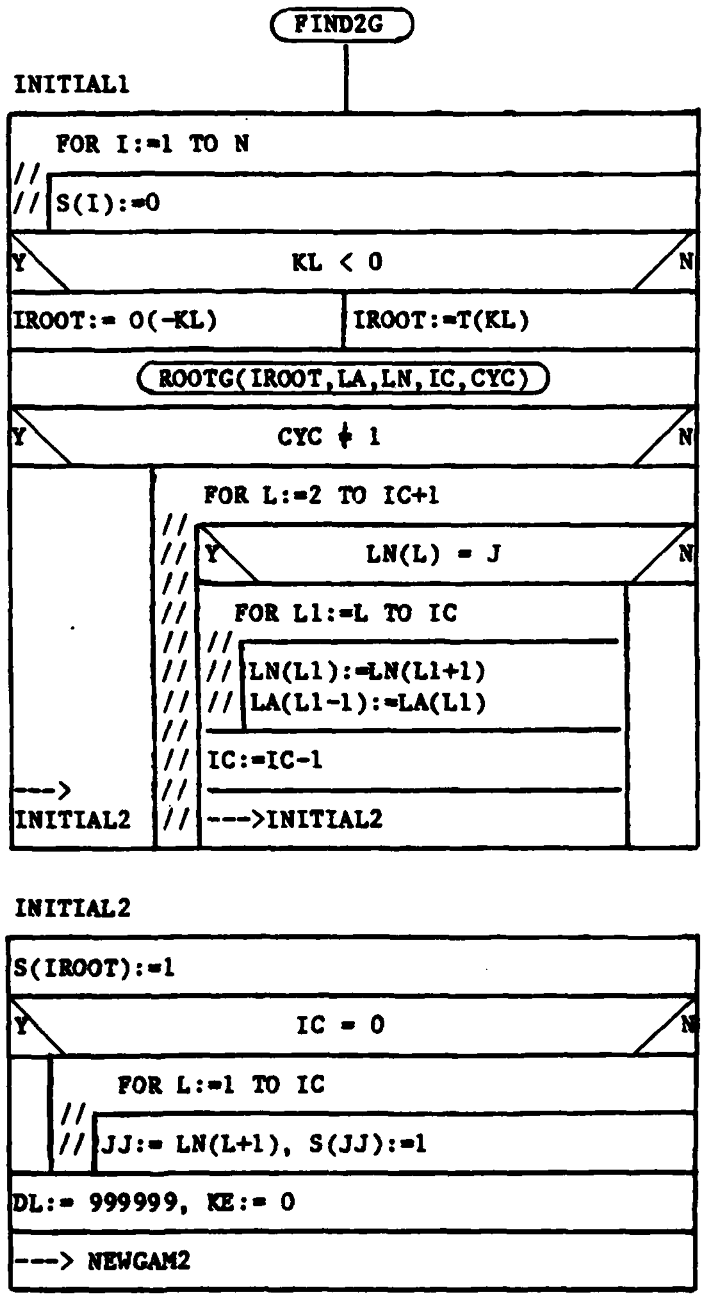
1.(INITIAL1) Set node set S(I)=0.If KL is a mirror arc find tree rooted at O(-KL). If not find tree rooted at T(KL). If tree rooted at IROOT forms a cycle, delete the second reference to node j from LN and delete KL from LA.

2.(INITIAL2) For each node in N2, set S(I) = 1. Initialize DL and KE.

3.(NEWGAM) Set γ for root node equal 1. Compute and assign γ's for each node in N2.

4.(SEARCH) Start search through list of arcs. If k is a basic arc or an artificial arc go to 8. Otherwise go to 5.

5.(DECIDE) Set I equal to the origin node of arc k. If k is a forward arc go to 6, otherwise go to 7.



```
FIND2G
INITIAL1
FOR I:=1 TO N
    S(I):=0
KL < 0
  Y: IROOT:= O(-KL)
  N: IROOT:=T(KL)
ROOTG(IROOT,LA,LN,IC,CYC)
CYC ≠ 1
  Y: --->INITIAL2
  N: FOR L:=2 TO IC+1
         LN(L) = J
           Y: FOR L1:=L TO IC
                  LN(L1):=LN(L1+1)
                  LA(L1-1):=LA(L1)
              IC:=IC-1
              --->INITIAL2

INITIAL2
S(IROOT):=1
IC = 0
  N: FOR L:=1 TO IC
         JJ:= LN(L+1), S(JJ):=1
DL:= 999999, KE:= 0
---> NEWGAM2
```

Subroutine FIND2G
Figure 5.14

6.(FORWARD) If KL is stored as a forward arc and KL is going to a lower bound (g(j) < 0), call GENER2. Otherwise call ABSORB. Go to 8.

7.(MIRROR) If KL is stored as a mirror arc and going to an upper bound (g(i) < 0), call GENER2. Otherwise call ABSORB. Go to 8.

8.(COUNT) When all the arcs have been checked, call PIVOT1G to pivot in KE, otherwise go to 4.

NEWGAM2

```
G(IROOT):=1
IC = 0
  Y:
  N:
    FOR L:=1 TO IC
        K:=LA(L), JJ:=LN(L+1)
        K > 0
          Y:
            II:=O(K)
            G(JJ):=G(II)*A(K)
          N:
            II:=T(-K)
            G(JJ):=G(II)/A(-K)
--->SEARCH
```

SEARCH

```
K = KL OR K = -KL
  Y:
    --->COUNT
  N:
    H(K) ≥ 99999
      Y:
        --->COUNT
      N:
        --->DECIDE
```

DECIDE

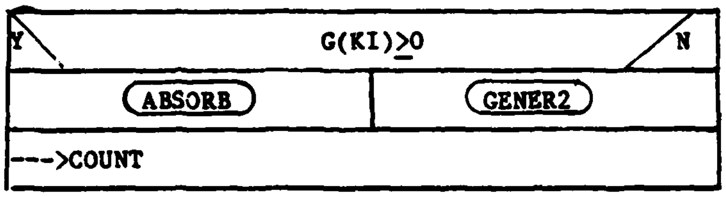
```
I:= O(K), J:= T(K)
KL > 0
  Y:
    KJ:= T(KL)
    --->FORWARD
  N:
    KI:= T(-KL)
    --->MIRROR
```

FORWARD

```
G(KJ) < 0
  Y:
    (GENER2)
  N:
    (ABSORB)
--->COUNT
```

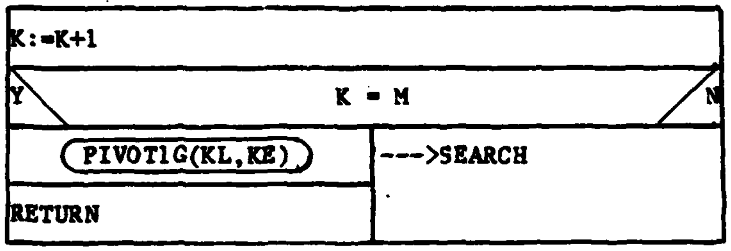
Subroutine FIND2G
Figure 5.14(cont)

MIRROR



```
G(KI)≥0
  Y: ABSORB
  N: GENER2
--->COUNT
```

COUNT



```
K:=K+1
K = M
  Y: PIVOT1G(KL,KE)
     RETURN
  N: --->SEARCH
```

Subroutine FIND2G
Figure 5.14(cont)

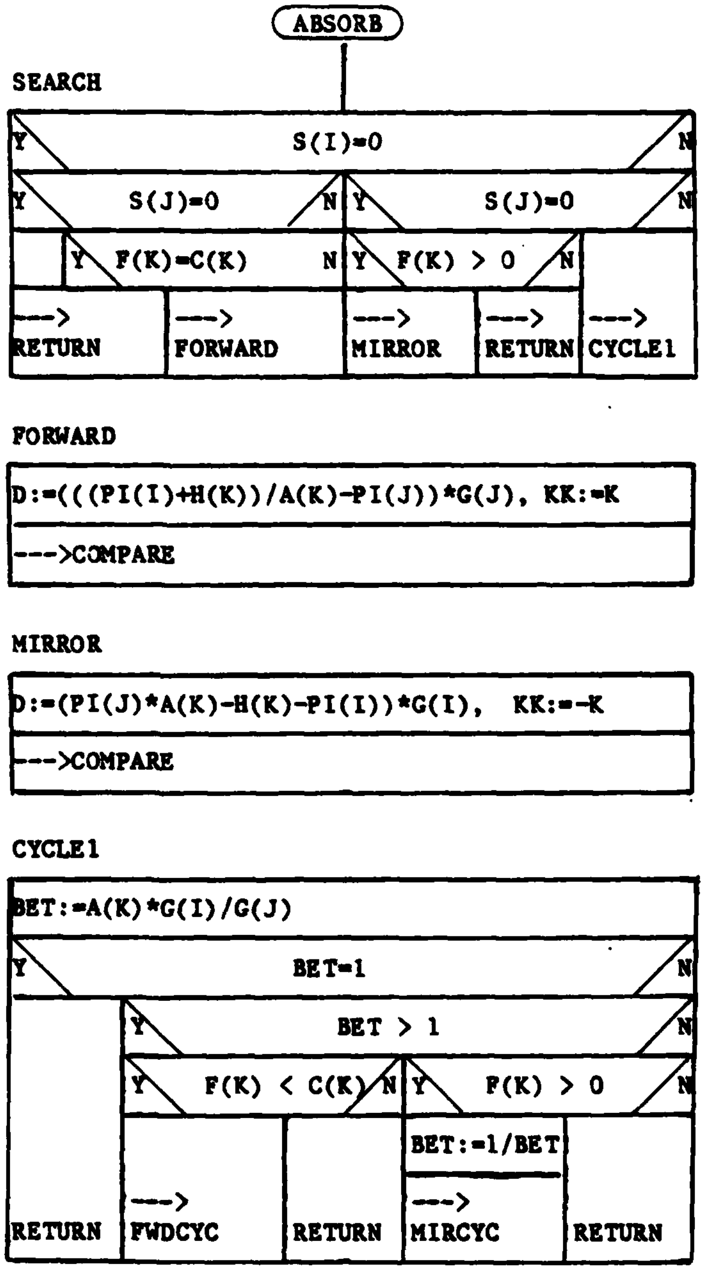
PURPOSE: To deter-
if arc k is admis-
sible to enter the
basis and calculate
the minimum dk
value.

1.(SEARCH) If arc
k originates in N1
(S(I)=0) and term-
inates in N2 (S(I)=
1), check forward
arc for admissibil-
ity. If admissible
go to 2. If arc k
originates in N2
and terminates in
N1, check mirror
arc for admissibil-
ity. If admissible
go to 3, otherwise
go to 4.

2.(FORWARD) Evalu-
ate dk for forward
arc and go to 7.

3.(MIRROR) Evaluate
dk for mirror arc
and go to 7.

4.(CYCLE1) Compute
gain of cycle by
inserting forward
arc. If β = 1, re-
turn. If gain > 1,
test forward arc
for admissibility.
If admissible, go to
5, otherwise re-
turn. Test mirror
arc for admissi-
bility. If admis-



Subroutine ABSORB
Figure 5.15

sible, go to 6,
otherwise return.

5.(FWDCYC) Compute dk for forward arc that forms a cycle. Go to 7.

6.(MIRCYC) Compute dk for mirror arc that forms a cycle. Go to 7.

7.(COMPARE) Test dk against smallest found to this point. If dk is smaller, call this the entering arc. Return.

FWDCYC

```
D:=(BET/(BET-1))*(((PI(I)+H(K)/A(K))-PI(J))*
   G(J)
KK:= K
--->COMPARE
```

MIRCYC

```
D:=(BET/(BET-1))*(PI(J)*A(K)-H(K)-PI(J))*G(I)
KK:=-K
```

COMPARE

```
Y                  D < DL                    N
DL:=D, KE:=KK      |
RETURN
```

Subroutine ABSORB
Figure 5.15(cont)

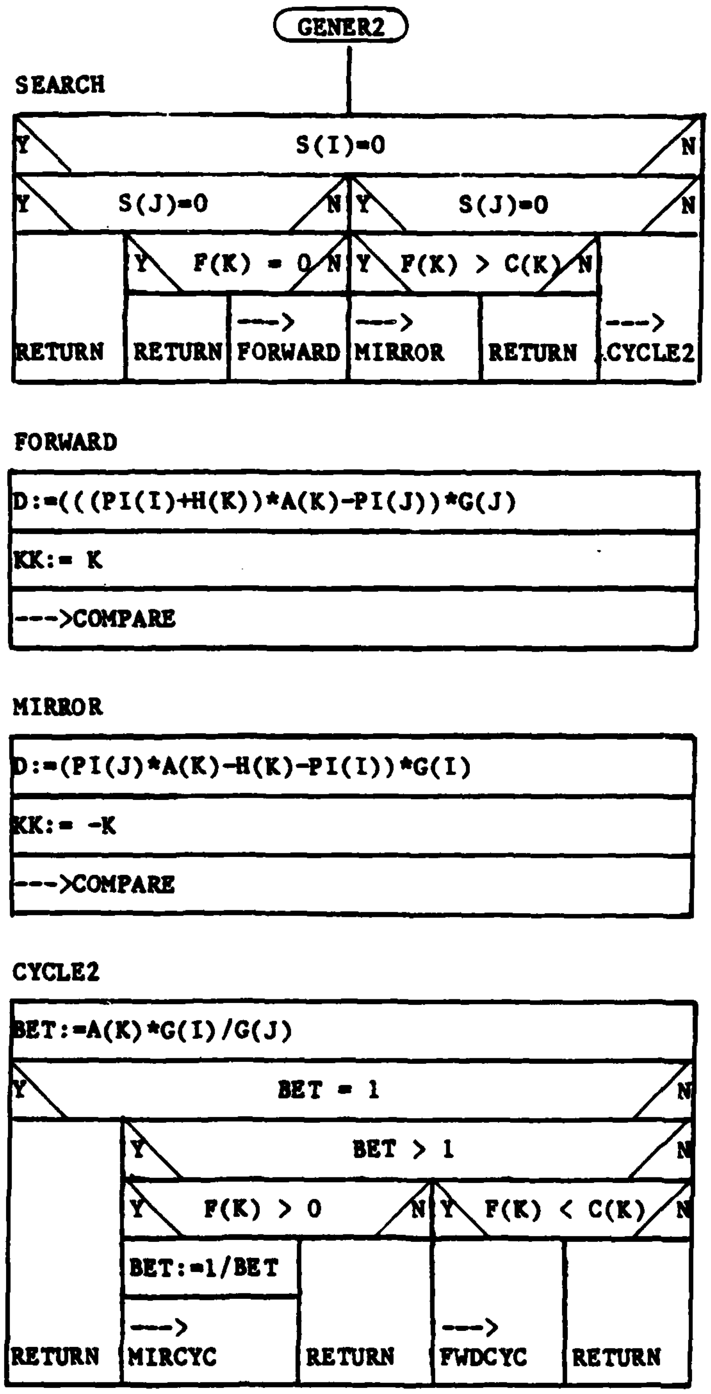
PURPOSE: To determine if arc k is admissible to enter the basis and calculate the minimum dk value.

1.(SEARCH) Check if arc k originates in N1 (S(I) =0) and terminates in N2 (S(I)=1). If so, check mirror arc for admissibility. If admissible go to 2, otherwise return. If arc k originates in N2 and terminates in N1, check forward arc for admissibility. If admissible go to 3, otherwise return. If arc k goes from N2 to N2, go to 4.

2.(FORWARD) Evaluate dk for forward arc. Go to 7.

3.(MIRROR) Evaluate dk for mirror arc. Go to 7.

4.(CYCLE2) Compute gain caused by adding arc k. If cycle gain > 1 and mirror arc is admissible, go to 6. If cycle gain is < 1 check forward arc for admissibility. If admissible go to 5. Otherwise return.



Subroutine GENER2
Figure 5.16

5.(FWDCYC) Compute dk for forward arc that forms a cycle. Go to 7.

6.(MIRCYC) Compute dk for mirror arc that forms a cycle. Go to 7.

7.(Compare) Test dk against smallest value to this point. If dk is smaller, call this arc the entering arc KE. Return.

```
FWDCYC
D:=(BET/(BET-1))*(((PI(I)+H(K))*A(K)-PI(J))*
   G(J)
KK:= K
--->COMPARE

MIRCYC
D:=(BET/(BET-1))*(PI(J)*A(K)-H(K)-PI(I))*G(I)
KK:= -K
--->COMPARE

COMPARE
Y            D < DL            N
DL:=D, KE:= KK
RETURN
```

Subroutine GENER2
Figure 5.16(cont)

$$d_{k_E} = \min_{k \in M_A} \{d_k\}. \tag{5.21}$$

This is the formula use in the subroutine COMPARE. It is clear that d_{k_E} is non-negative because the process begins with a dual feasible solution.

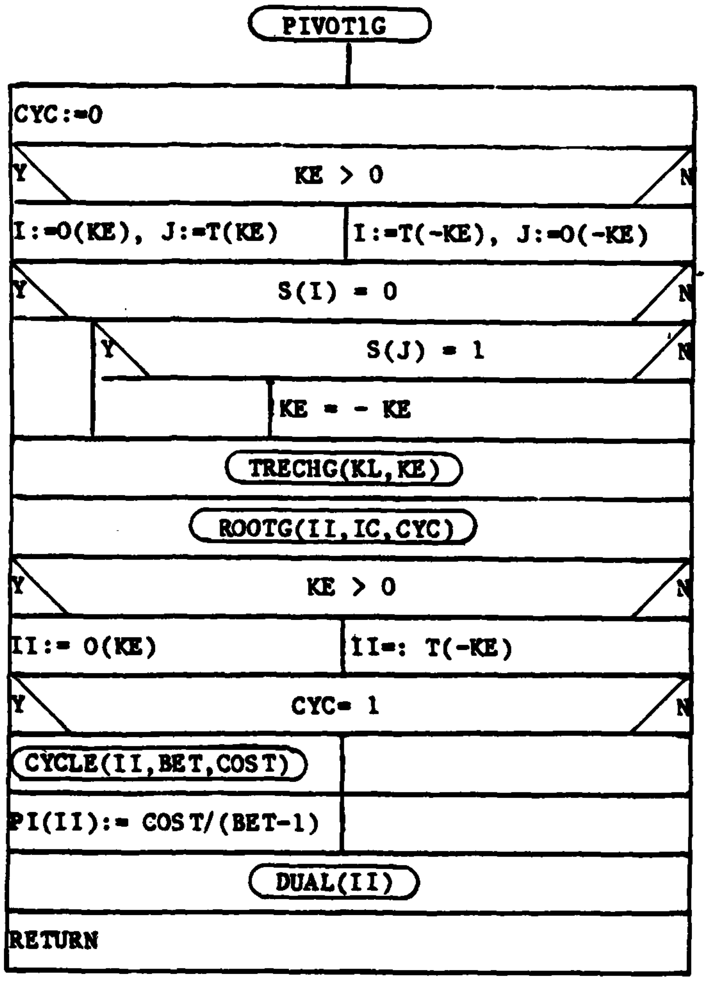
The concepts developed above are implemented in the subroutines FIND2G, ABSORB, and GENER2.

5.8 A Basis Change Method

At each iteration subroutine MFLOG2 will determine the arc that limits the flow change on the flow augmenting trail(s), and that arc will leave the basis. Subroutine FIND2G will determine the entering arc. Arc k_E is such that $k_E \in (N_1,N_2)$, $k_E \in (N_2,N_2)$, or $k_E \in (N_2,N_1)$. In the first two cases the subroutine TRECHG [pg 116,(46)] can be used directly to accomplish the basis change operation for the PSENG algorithm. However, if $k_E \in (N_2,N_1)$, k_E must enter as $-k_E$ as TRECHG assumes k_E's origin is in the set N_1.

In order to update the dual variables, π , the set of nodes and arcs in the tree rooted at j_{k_E} is found with the subroutine ROOTG [pg.247, (46)].

PURPOSE: To change the basis tree by adding KE and deleting KL. Compute new π values for new basis.



Subroutine PIVOT1G
Figure 5.17

If k_E has formed a cycle, subroutine CYCLE [pg.273, (46)] is used to calculate the gain of the cycle formed by k_E. The π values are then updated with the subroutine DUAL [pg.271, (46)]. These concepts are implemented in the subroutine PIVOT1G.

5.9 The Complete Algorithm

At this point we can state the complete algorithm used in PSENG.

1. Solve the original problem to optimality by any primal or dual solution technique.
2. Determine the list of candidate arcs and the list of parametric parameters.
3. Find the flow augmenting trail or trails which are generated by the candidate arcs. Determine the arc to leave the basis. If none exists, go to step 4. If the leaving arc is a non-basic candidate arc, go to step 5. Otherwise go to step 6.
4. At least one basic candidate arc can have its capacity increased without bound indicating no further parametric changes

will affect the basis. Stop.

5. Augment the flow as much as possible on the trail or trails. Adjust the capacity and flows on the candidate arcs. Stop, any futher iterations would cause infeasibility.
6. Augment the flow as much as possible through the trail or trails. Adjust the capacity and flows on the candidate arcs.
7. Find an arc to enter the basis.
8. Change the basis by deleting the leaving arc and inserting the entering arc. Modify the dual variables to satisfy complementary slackness. Return to step 3.

It is important to remember that for any gains algorithm that the arithmetic is performed with real numbers rather than with integers. Thus the problem of computer round-off error is quite likely to occur. The tests for equality must take this into account. For example in the subroutine ABSORB, the test $f_k > 0$ is replaced by $f_k > \varepsilon$,where ε is a small quantity. While the flow charts presented in this section and in previous sections do not

PURPOSE: To implement the parametric analysis for a generalized network.

1.(INITIAL) Set all intial arrays and counters to zero.

2.(Remember) Store all values from the optimal solution, which allows the optimal solution to be recovered.

3.(QUESTION) Input list of candidate arcs and parametric parameters on which the parametric analysis is to be accomplished.

```
PSENG
INITIAL
    FOR I:=1 TO N
        XP(I):=0,XX(I):=0,XB(I):=0,XR(I):=0,
    FOR I:=1 TO M
        AH(I):=0,XL(I):=0,PP(I):=0,RC(I):=0,
        XK(I):=0,MF(I):=0,XC(I):=0,XF(I):=0,
        P(I):=0
    K:=0, ST:=0, MX:=0, ITER:=0
REMEMBER
    FOR I:=1 TO N
        XP(I):=PI(I),XR(I):=PR(I),XB(I):=PB(I),
        XX(I):=PF(I)
    XX:=HH,:HH:=0
    FOR I:=1 TO M
        XF(I):=F(I),XC(I):=C(I),HH:=HH+(F(I)+CL(I))
        *H(I)
    AH(0):=HH, XL(0):=0,MF(0):=0,C(0):=0
QUESTION
    INPUT XM
    FOR I:=1 TO XM
        INPUT XK(I), K:=XK(I),
        INPUT PP(I),P(K):=PP(I)
    --->CHECK
```

Main Program PSENG
Figure 5.18

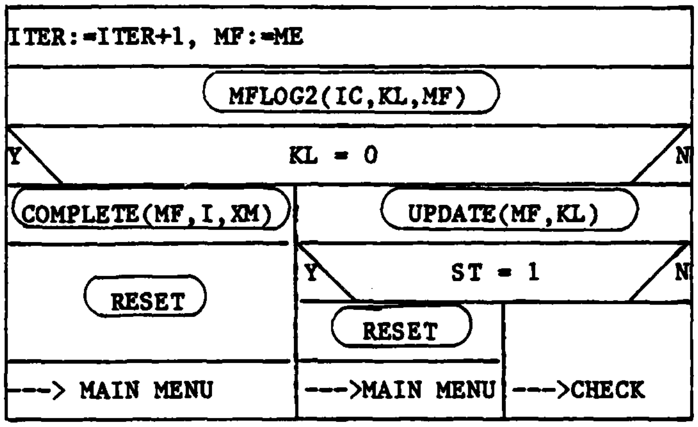
4.(CHECK) Initialize Υ's to zero and node check values to zero. For each candidate arc check if the arc is nonbasic and its parametric parameter is negative. If so calculate the reference number. Store smallest reference number as ME. If parametric parameter is negative set IE to the terminal node of XK otherwise let IE be the origin node. Call PATH which will create LA, LN, and RC lists for all flow augmenting trails created by the nonbasic candidate arcs. If no cycles are found (z=0) go to 5. Otherwise call CYCLEG to update Υ's for affects of the cycle.

```
CHECK
FOR I:=1 TO N
    G(I):=0,IK(I):=0
IC:=0,CK:=0,Z:=0,ME:=999999,NB:=0,NK:=0
FOR I:=1 TO XM
    XK:=XK(I),ZI:=O(XK),ZJ:=T(XK)
    Y  PB(ZJ) = XK  OR  PB(ZI) = -XK  N
        Y  P(XK) < 0  N
            Y: M1:=C(XK)/-P(XK)
               P:-P(XK),IE:=T(XK)
               JE:=O(XK),KE:=-XK
               Y  M1 < ME  N
                   Y: ME:=M1,NK:=XK
            N: N  F(K) = 0  Y
                   N: P:=P(XK),IE=O(XK)
                      JE:=T(XK),KE:=XK
            IJ:=IE, MULT:=1
            (PATH(IJ,MULT,LN,LA,RC,CK))
            IJ:=JE,MULT:=-1
            (PATH(IJ,MULT,LN,LA,RC,CK))
I:=1
Y  Z = 0  N
    N: (CYCLEG)
--->AUGMENT
```

Main Program PSENG
Figure 5.18(cont)

5.(AUGMENT) Call MFLOG2 to determine the maximum reference number for the basic arcs. If no basic arc can be found to leave the basis, call COMPLETE and return to MAIN MENU. Otherwise call UPDATE to change arc flows and candidate arc capacities. If in update a stopping condition is reached (st=1), call RESET and return to MAIN MENU. Otherwise go to 4.

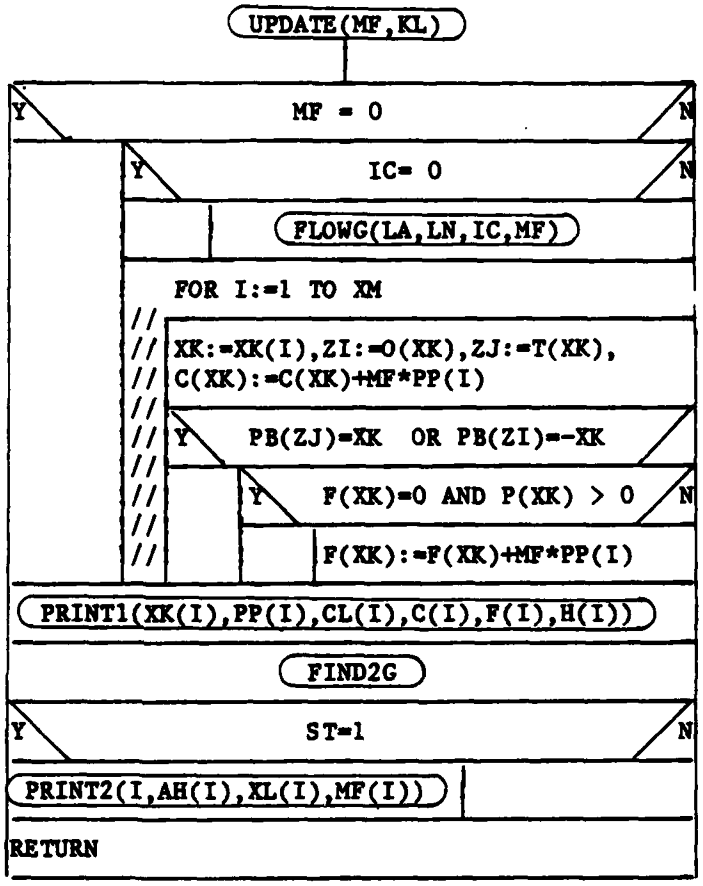
AUGMENT



Main Program PSENG
Figure 5.18(cont)

PURPOSE: To update flows and capacities on candidate arcs and to update flows on appropriate basic arcs. Also to find KE to enter basis, and to print out intermediate results.

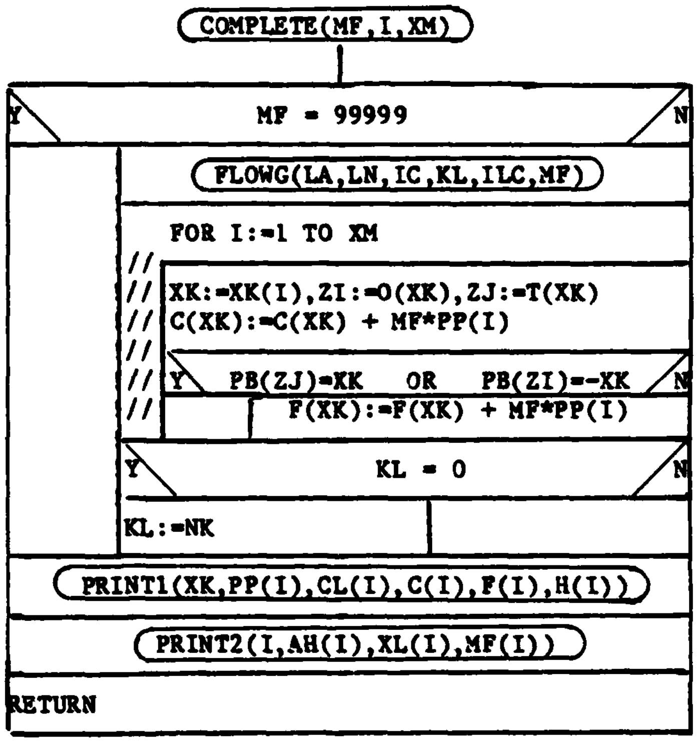
If MF = 0, this is a degenerate iteration. Otherwise update flows for the basic arcs found in LA. Update the flows and capacities for the candidate arcs. Then print iteration results. Call FIND2G to determine KE, the entering arc. If a stopping condition has been found call PRINT2 to print final results, otherwise return.



Subroutine UPDATE
Figure 5.19

PURPOSE: Final update of flows on basic arcs and to update capacities on candidate arcs. Print final iteration results and summary table.

Note: If MF = 99999, a basic arc can have its capacity increased without bound.



```
COMPLETE(MF,I,XM)
MF = 99999
  Y:
  N:
    FLOWG(LA,LN,IC,KL,ILC,MF)
    FOR I:=1 TO XM
      XK:=XK(I),ZI:=O(XK),ZJ:=T(XK)
      C(XK):=C(XK) + MF*PP(I)
      PB(ZJ)=XK   OR   PB(ZI)=-XK
        Y:
        N:
          F(XK):=F(XK) + MF*PP(I)
    KL = 0
      Y:
        KL:=NK
      N:
PRINT1(XK,PP(I),CL(I),C(I),F(I),H(I))
PRINT2(I,AH(I),XL(I),MF(I))
RETURN
```

Subroutine COMPLETE
Figure 5.20

reflect such adjustments, these adjustments do appear in the computer implementation of PSENG. The concepts presented above are implemented in the algorithm PSENG.

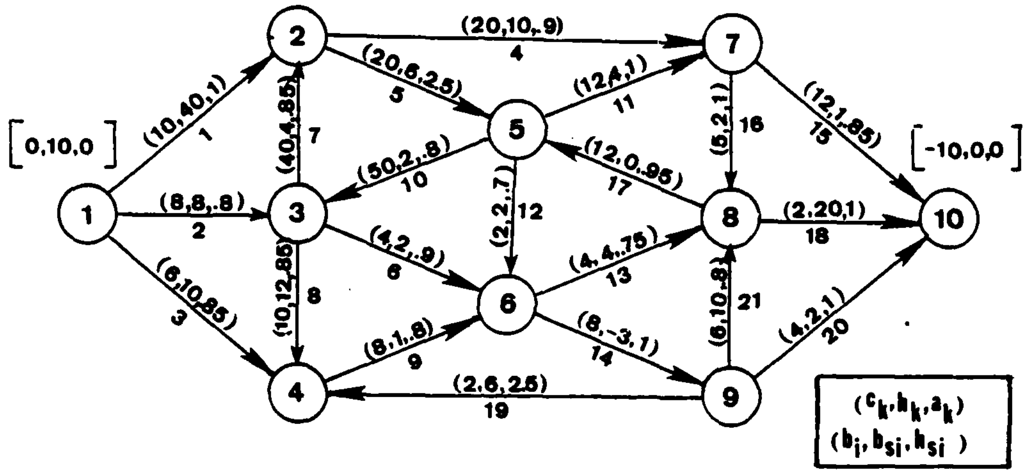
Figure 5.21 presents an example application of the algorithm PSENG. Figure 5.21(a) lists all the initial parameters of the problem and Figure 5.21(b) shows the optimal basic feasible solution derived by a primal solution method. In each of the Figures 5.21(c) through 5.21(h), the entering arc is indicated by a dashed line and the leaving arc is marked by an X. The basic arcs are indicated by heavy lines and the non-basic arcs at capacity by light lines. The dashed-dot lines indicate non-basic candidate arcs with flows at capacity. The flow and capacity at the completion of each basis change and the resulting total cost of those flows are also given at each step.

The candidate arc list and their respective parametric parameters are listed below

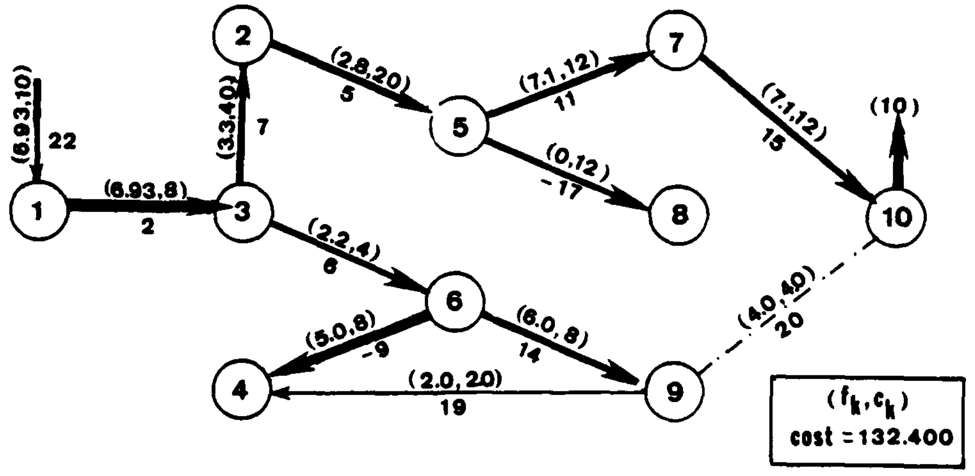
$$X = [5,6,7,11,14,15,20]$$

$$P = [-.3,1.2,-2.1,-.6667,.1,-3,1.4].$$

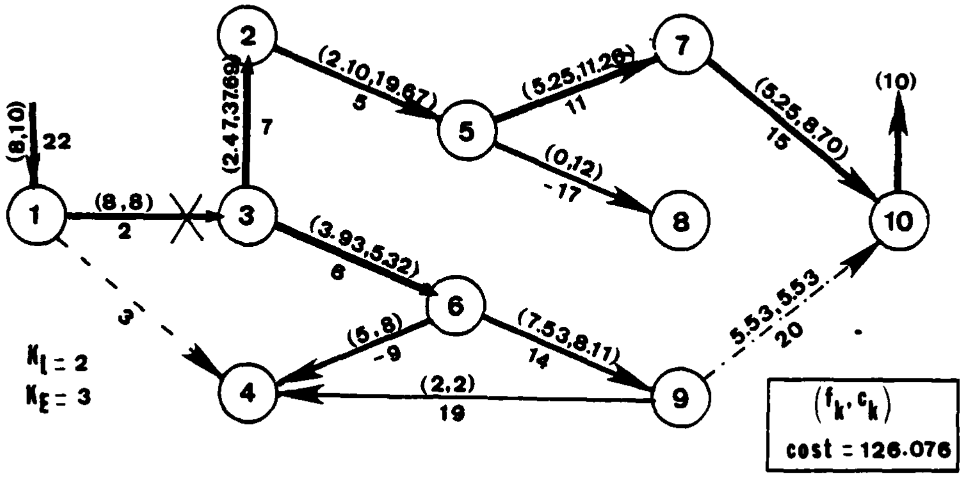
In Figure 5.21(c) candidate arc 20 creates two flow augmenting paths. The values of the γ's are computed in PATH, and MFLOWG2 calculates a reference number of 1.10. Arc 2 is forced to leave the



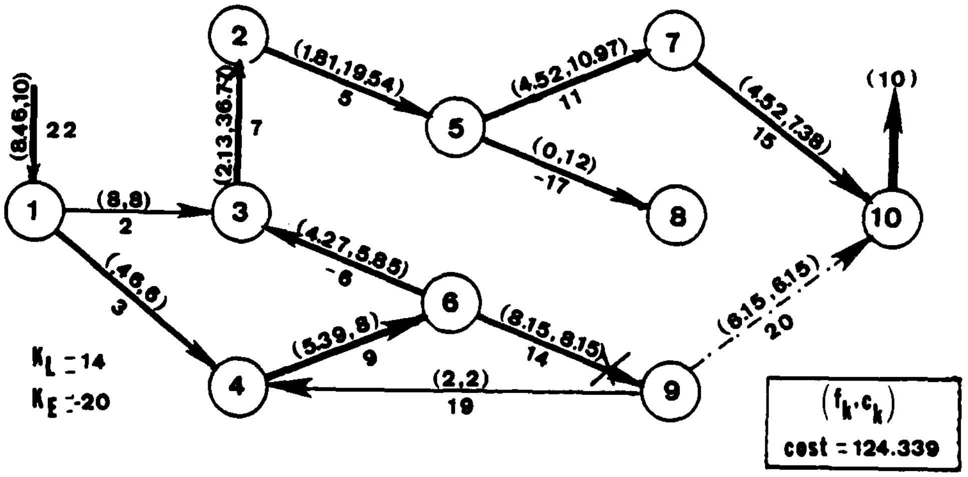
(a)
Example of PSENG
Figure 5.21



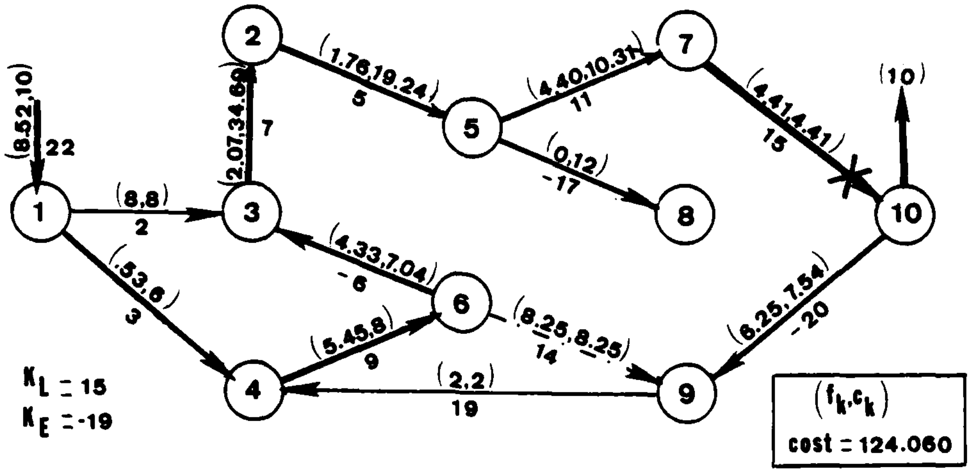
(b)
Example of PSENG
Figure 5.21(cont)



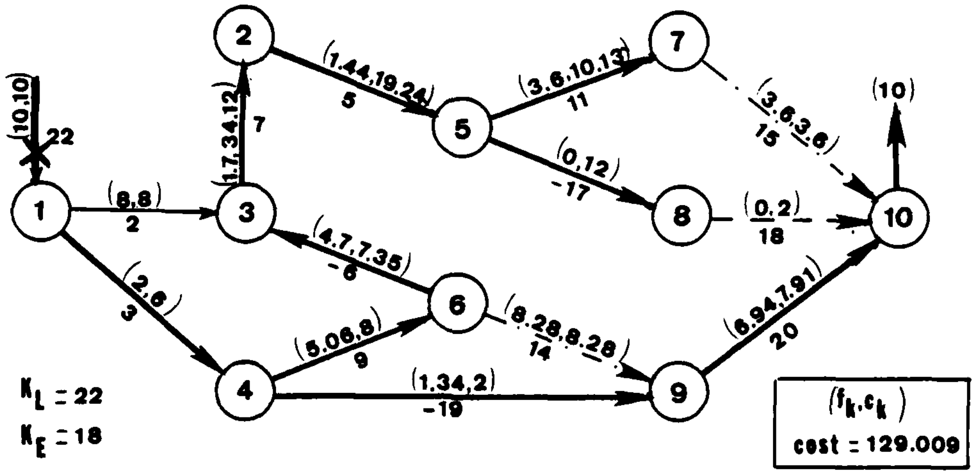
(c)
Example of PSENG
Figure 5.21(cont)



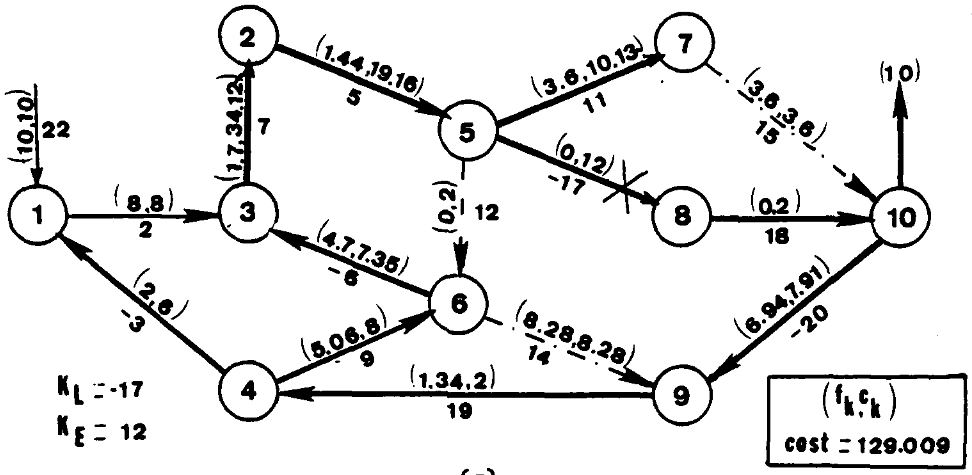
(d)
Example of PSENG
Figure 5.21(cont)



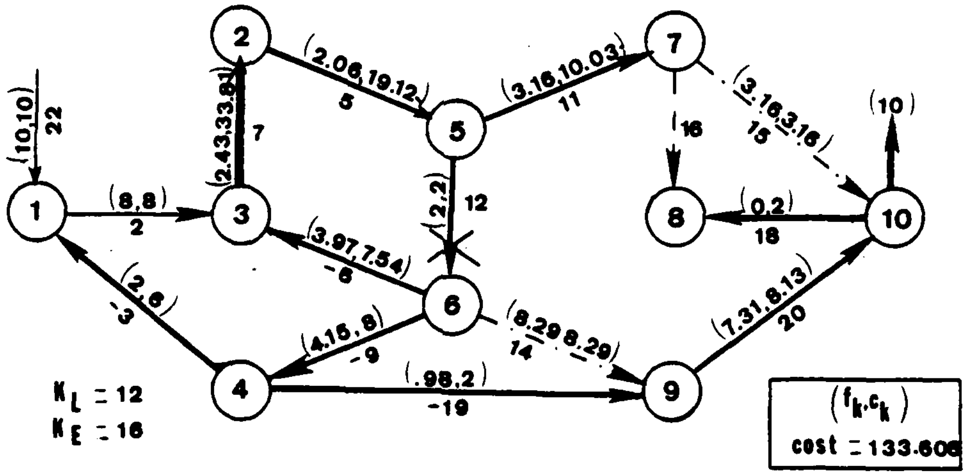
(e)
Example of PSENG
Figure 5.21(cont)



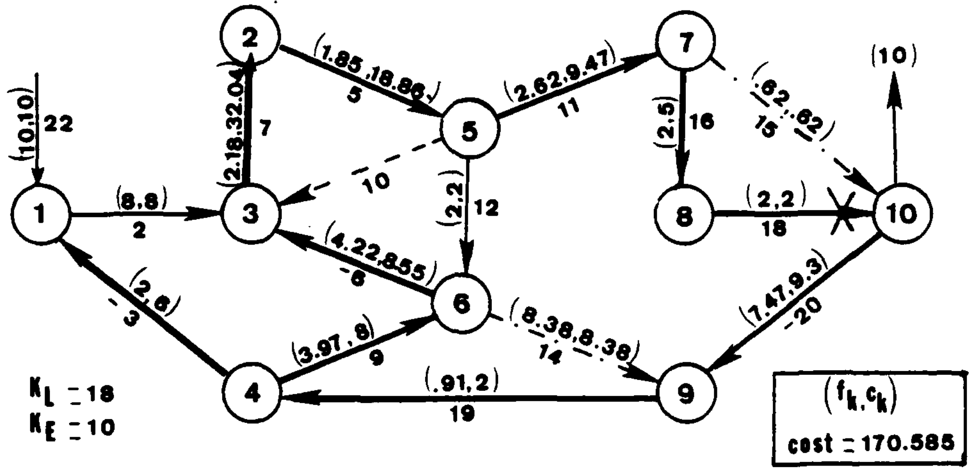
(f)
Example of PSENG
Figure 5.21(cont)



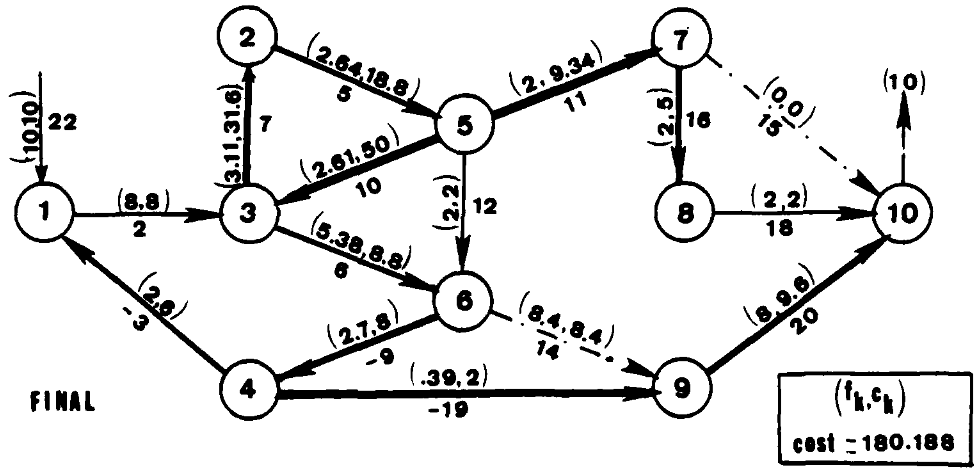
(g)
Example of PSENG
Figure 5.21(cont)



(h)
Example of PSENG
Figure 5.21(cont)



(i)
Example of PSENG
Figure 5.21(cont)



(j)
Example of PSENG
Figure 5.21(cont)

basis at its upper bound. The subroutine UPDATE calls FIND2G which determines the entering arc, $k_E = 3$, and updates the flows and capacities on the candidate arcs and the flows on the basic arcs on the flow augmenting trails.

Figure 5.21(f) shows four augmenting trails: two generated by non-basic candidate arc 14 and two generated by non-basic candidate arc 15. Figure 5.21(h) shows a cycle which is formed when arc 12 enters the basis. The algorithm terminates when the flow and capacity of candidate arc 15 are driven to zero. This is shown in Figure 5.21(j).

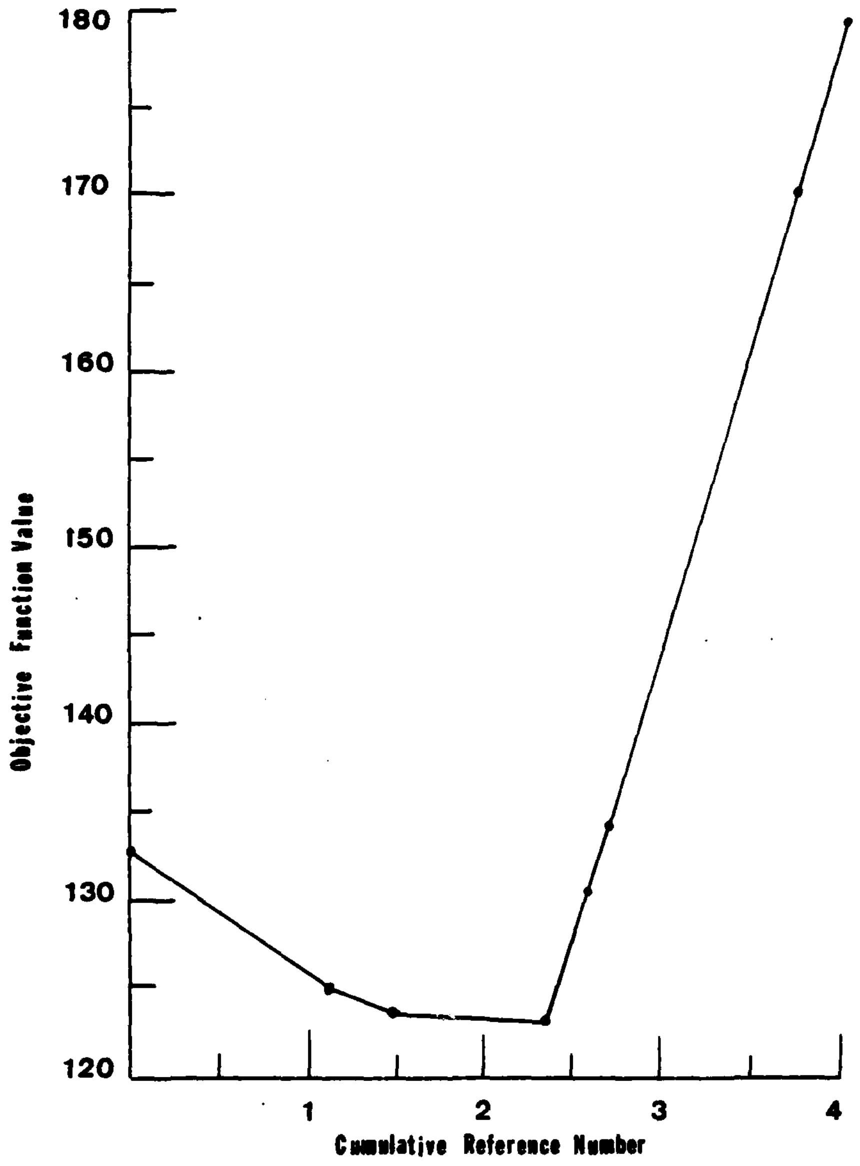
Table 5.2 summarizes the results of the algorithm. Figure 5.22 illustrates that a parametric analysis on the capacities of a network produces a piecewise-linear and convex function. It is interesting to note that the objective function decreases until the third iteration, when the objective function begins to increase.

5.10 Extensions to Node External Flows

In sections 5.2 through 5.9 we examined the effects of parametric analysis on the arc capacity vector. In the context of linear programming, we have only analyzed the upper bound part of the right-hand side vector. We can easily extend the results of sections 5.2 through 5.9 to accommodate the effects of changing the

ITERATION NUMBER	OBJECTIVE FUNCTION VALUE	LIMITING ARC	CUMULATIVE REFERENCE NUMBER
0	132.400	-	-
1	126.076	02	1.096
2	124.339	14	1.538
3	124.060	15	2.530
4	128.009	22	2.798
5	128.009	17	2.798
6	133.606	12	2.847
7	170.585	18	3.792
8	180.188	15	4.000

Summary Table of PSENG Example Problem
Table 5.2



Optimal Objective Function Value as a Function of the Cumulative Reference Number
Figure 5.22

supply and demand values expressed by the right-hand side vector b. While previously we focused our attention on changing arc capacities, we will now show how to extend the PATH subroutine to take into consideration changes in the external flow.

Because PATH starts at a node and traces the flow augmenting path back to the root node or a root cycle, we merely need to designate a parametric parameter for each node at which the external flow is to be changed. Then PATH will determine the Υ's for each node generated by these flow augmenting trails. Subroutine MFLOWG2 will determine the limiting arc. We then need only determine the entering arc in FIND2G. We continue this process until any further changes will cause the problem to become infeasible or the current basis will remain optimal for any further changes.

CHAPTER 6

APPLICATIONS OF PARAMETRIC SENSITIVITY ANALYSIS

6.1 Introduction

In this chapter we develop two dual incremental algorithms which use the concepts of parametric analysis developed in Chapter 5. The first algorithm is called PARARC which uses parametric analysis on arc capacities. The second algorithm is called PARANDE which uses parametric analysis on node external flows. The computational efficiency of PARARC and PARANDE are compared against a dual-incremental code called INCREMG (46) and two primal codes, NETG (38) and PGAINS (46). Lastly we discuss applications of parametric analysis to other classes of problems.

6.2 Parametric Arc Incremental Flow Algorithm

Given a network with arcs having capacity, cost, and gain parameters with source and destination nodes defined and with a required quantity of flow specified at the sources and destinations, our problem is to determine the arc flows with minimum total cost.

The problem can be stated as

$$\text{Min} \quad \sum_{k=1}^{m} h_k f_k \tag{6.1}$$

$$\text{S.T.} \quad -\sum_{k \in M_{Ti}} a_k f_k + \sum_{k \in M_{Oi}} f_k = b_i \quad \text{for all } i \in N-s \tag{6.2}$$

$$0 \leq f_k \leq c_k \quad k \in M \ . \tag{6.3}$$

The parametric arc incremental algorithm is a dual-node infeasible approach applied to a special network. The network has a single super-sink (node t) and a single super-source (node s).

For each source node an arc is created from the super-source to the source node with capacity equal to b_i. For each destination node an arc is creat·d from the destination node to the super-sink with capacity equal to $-b_i$. We let the expanded arc set which is created be denoted as M'. We can redefine the primal problem as

$$\text{Min} \quad \sum_{k=1}^{m} h_k f_k \tag{6.4}$$

$$\text{S.T.} \quad -\sum_{k \in M_{Ti}} a_k f_k + \sum_{k \in M_{Oi}} f_k = 0 \quad i=1,\dots,n \quad i \neq s,t \tag{6.5}$$

$$-\sum_{k \in M_{Tt}} a_k f_k + \sum_{k \in M_{Ot}} f_k = -F_t \tag{6.6}$$

$$0 \leq f_k \leq c_k \quad k \in M' \ . \tag{6.7}$$

Here F_t is the sum of the destination demands. An assignment of flows that satisfy (6.5) and (6.7) will be called F. An F which satisfies (6.4) for some value of flow at the sink less than F_t will be called an intermediate optimum. The flow F that satisfies (6.4-6.7) is called the optimum.

To solve this problem using the parametric approach of Chapter 5, we identify the arcs from the destination nodes to the super-sink as candidate arcs with parametric parameters equal to the demand at the destination. We then increase the candidate arc capacities from zero until they reach a capacity equal to the original destination demands. This creates multiple flow augmenting paths at each iteration and allows flows to be augmented from the super-source to the super-sink.

Thus we begin with an intermediate optimum F_0 for some value of output flow at the super-sink. Next we obtain a shortest path network D_s^0, that determines for each node the minimum cost per unit of additional flow to the node and the path over which the flow may be obtained. The output flow is increased in D_s^0 along the flow augmenting trails generated by the candidate arcs until one or more arcs in the trails become capacitated. This flow augments F_0 to become F_1. Then F_1 is the intermediate optimum. A new augmenting network is constructed, D_s^1, and the output flow is again augmented to obtain F_2. The process continues until the flow at the super-sink equals F_t. At every step F_k is an intermediate optimum; therefore at the termination the flow pattern must be at the optimum.

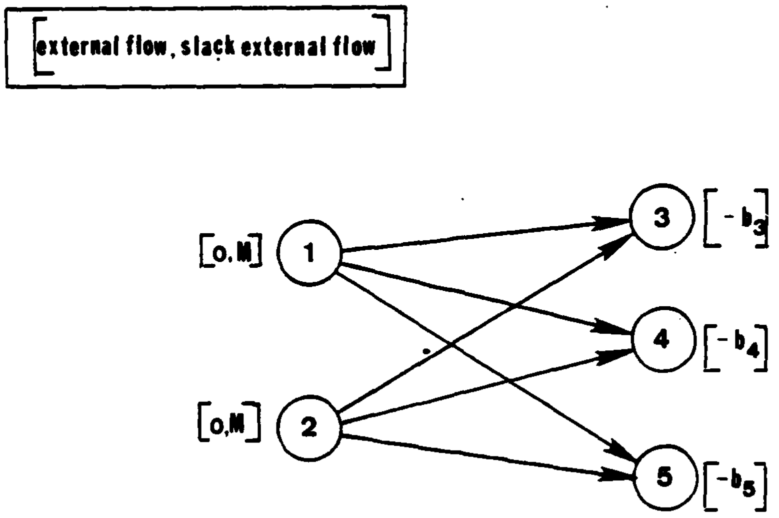
The general steps of the algorithm are as follows:

1. Find an initial least cost spanning tree.
2. If the flow at the super-sink is F_t, stop with the optimal solution.
3. For each candidate arc find the flow augmenting trails which include the basic arcs. Determine the maximum allowable flow increment which will occur on one of the trails. Determine the leaving arc. Augment the flows.
4. Find an arc to enter the basis. If none exists, stop. There is no feasible solution. Otherwise go to 5.
5. Change the basis by deleting the leaving arc and inserting the entering arc. Modify the dual variables to satisfy complementary slackness. Return to step 2.

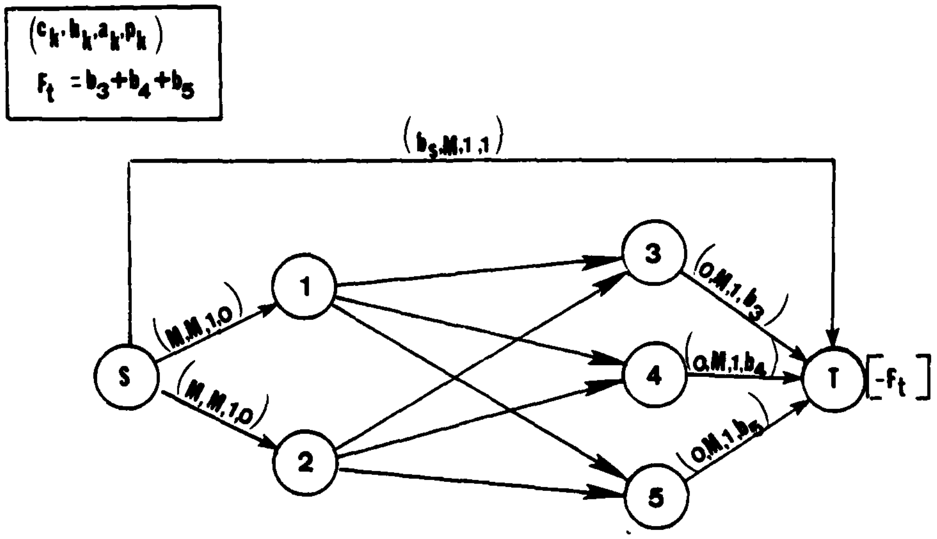
We now will develop the detailed steps of the algorithm.

6.2.1 Obtaining the Initial Basis Tree

In order to apply the concepts developed in Chapter 5, we must modify how the network is to be represented. Consider the sample network problem shown in Figure 6.1. We will create a



Sample PARARC Problem
Figure 6.1



The Augmented Network
Figure 6.2

super-sink node, which will have a fixed external flow equal to the sum of the destination fixed external flows. We also will create "candidate arcs" from the destination(s) to the super-sink with the following parameters:

$c_k = 0$ (capacity)

$h_k = M + 1$ (cost)

$p_k = - \{b_i [o(k)]\}$ (parametric parameter)

$a_k = 1$ (gain).

We will also create arcs from the slack node to all nodes with slack external flows with the arc parameters

$c_k = b_{si}$

$h_k = 0$

$a_k = 1$

$p_k = 0$.

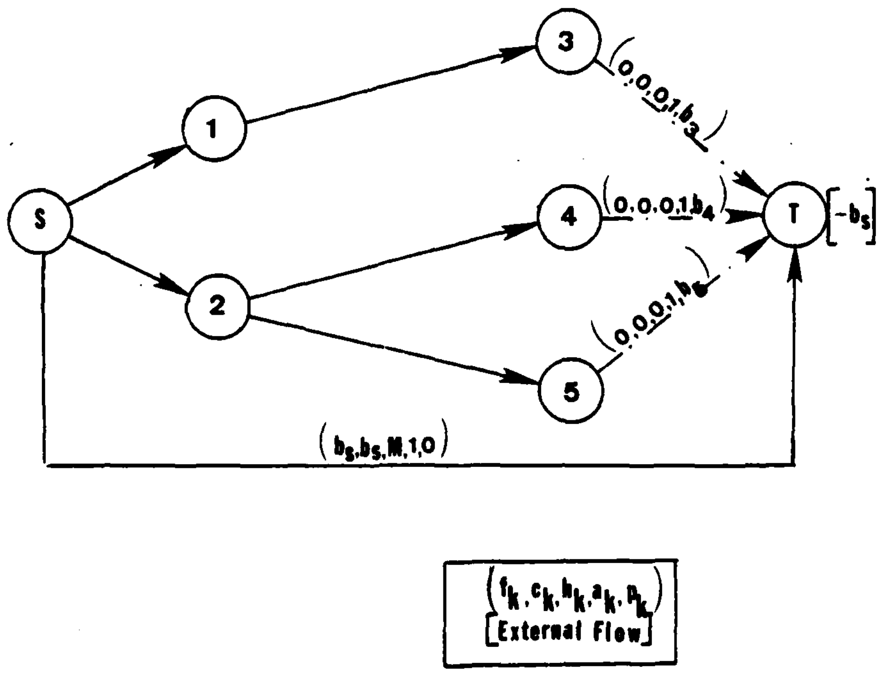
Finally we create an artificial arc from the slack node to the super-sink with parameters

$f_k = -F_t$

$$c_k = -F_t$$
$$h_k = M$$
$$a_k = 1$$
$$p_k = 0 .$$

Since PSENG requires conservation of flow at all nodes, an artificial arc is required to obtain conservation of flow at the super-sink. When the flow on the artificial arc goes to zero, the flow through the network will be F_t. All of these transformations are accomplished in the subroutines READG1 and ORIGSG1.

Once the transformed network is obtained, we use Dijkstra's algorithm adapted for the generalized network, as documented in [46,page 283], to obtain an initial basic feasible solution. The subroutine TREINT [46,page 121] is used to initialize the parameters of the basis tree. Since a generalized network flow problem may have arc gains greater than one or have negative arc costs, Dijkstra's algorithm may not provide an optimal shortest path tree. Therefore, we now use the algorithm PSHRTG [46,page 285] to find the initial least-cost spanning tree. Once this tree is found we reset the costs on the "candidate arcs" to be zero. This is illustrated in Figure 6.3. We reset these costs so when the candidate arc flows increase from zero to the former external flows of the original destination nodes, the total cost for the solution will reflect the



The Initial Basis
Figure 6.3

actual cost for the problem. At this point we have an initial optimal (all $d_k \geq 0$), basic, infeasible (the artificial arc has flow > 0) solution.

6.2.2 Determining the Arc to Leave the Basis

Similarly as in PSENG, we now use the PATH subroutine to determine the flow augmenting trails for each candidate arc. This will create a list of arcs (LISA), a list of nodes (LISN), and a list of gamma values (G). Using these lists MFLOG [46,page 266] will determine the maximum flow change or reference number on these trails and the limiting arc. We will denote the limiting arc as k_L.

6.2.3 Determining the Entering Arc

We use the subroutine FINDARC to determine the entering arc, k_E. If no arc can be found to enter the basis at this iteration, no feasible solution exists to the proposed problem.

6.2.4 Changing the Basis

Updating of the basis tree representation and the dual variables is accomplished in the subroutine PIVOTIG.

6.2.5 The Complete Algorithm

We can now give a complete statement of the parametric arc incremental algorithm.

1. Use a variant of Dijksta's shortest path algorithm to find the initial least-cost spanning tree. Use the arcs in the spanning tree to define a set of node labels π_i such that $\pi_s = 0$ and $\pi_j = (\pi_i + h_k) / a_k$ for all arcs $k(i,j) \varepsilon M_T$.

2. Use a triple labeling scheme to label and store the least-cost spanning tree.

3. If the flow on the artificial arc from the super-source to the super-sink node is zero, stop with the optimal solution.

4. For each candidate arc from a destination node to the super-sink, find the augmenting trails which include the candidate arcs and

the basic arcs. Determine the maximum allowable flow increment on these trails. Determine the arc on these trails which is at capacity and designate it the leaving arc k_L. Augment the flows.

5. Delete arc k_L from the spanning tree, partitioning the network into two subtrees defined by the sets N_1 and N_2. Search the set of admissible arcs originating or terminating in N_1 which connect to the set N_2 or the admissible arcs that originate and terminate in N_2. If no admissible arc exists, stop; there is no feasible solution. Otherwise select the arc with the minimum d_k and designate it as the entering arc k_E.

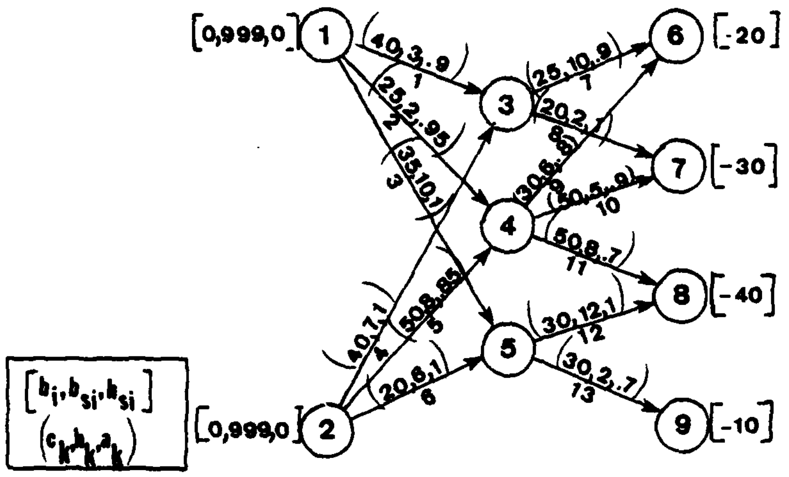
6. Perform a basis change which determines the new least-cost spanning tree. The basis change is accomplished by deleting k_L and adding k_E. Update the dual variables and return to step 2.

The algorithm PARARC implements these concepts. The flow charts for the FORTRAN coding of this algorithm can be found in Appendix 1.

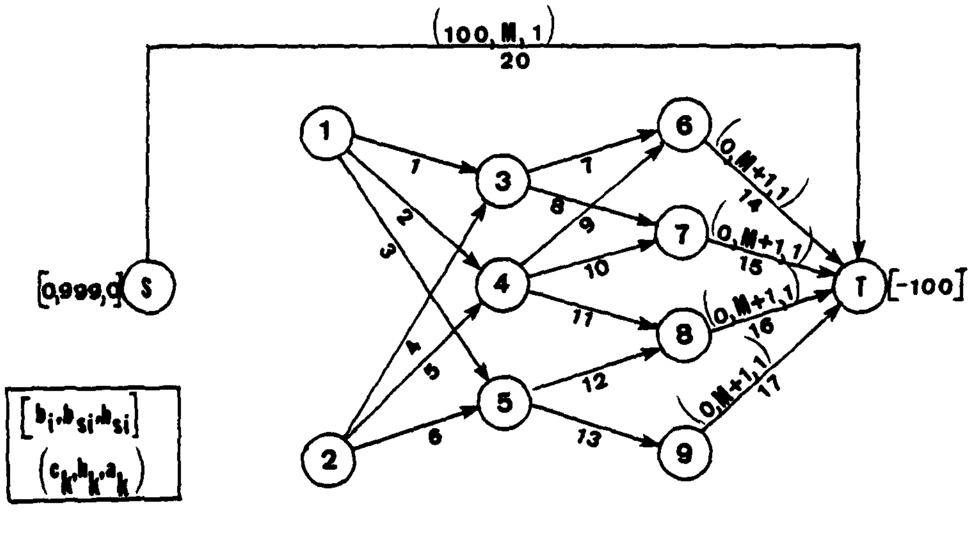
Figure 6.4 presents an example application of the algorithm PARARC. Figure 6.4(a) shows the initial problem data. Figure 6.4(b) illustrates the intermediate transformation which creates the four candidate arcs from the original destinations to the super-sink and the creation of the artificial arc from the slack node to the super-sink.

In each of the Figures 6.4(c) through 6.4(k) the entering arc is indicated by a dashed line and the leaving arc by an X. The basic arcs are indicated by heavy lines and the nonbasic arcs at capacity by light lines. The dashed-dot lines indicate the non-basic candidate arcs. The flow and capacity at the completion of each basis change are given at each step.

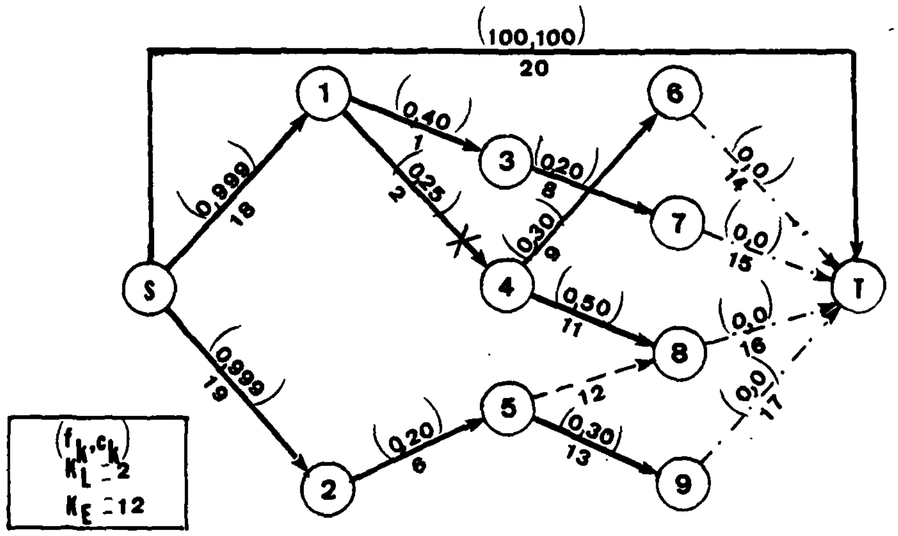
Figure 6.4(c) shows the initial least-cost spanning tree derived from a variant of Dijstra´s algorithm. Figures 6.4(d) through 6.4(j) show the intermediate steps in the solution process as the flow on the artificial arc is being decreased. It is interesting to note at each intermediate iteration, we are increasing the flows on multiple flow augmenting paths, rather than a single path as in the usual flow augmentation approach (46).



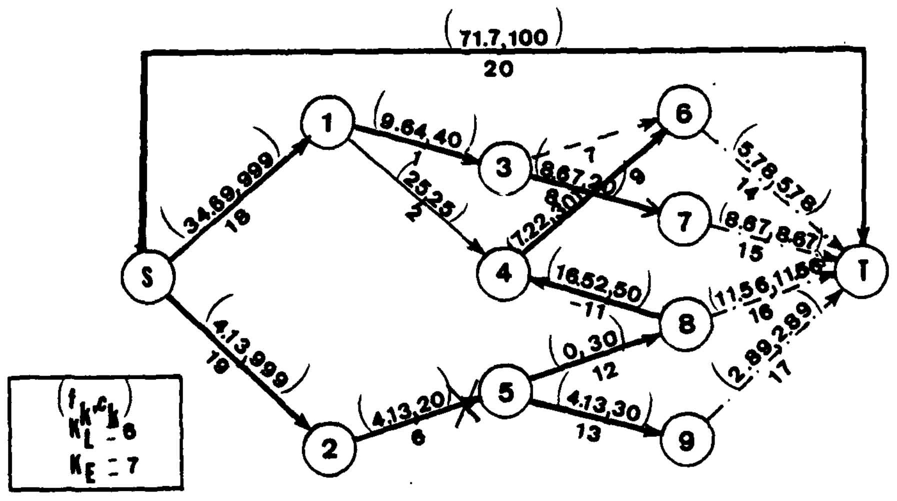
(a)
Example Application of PARARC Algorithm
Figure 6.4



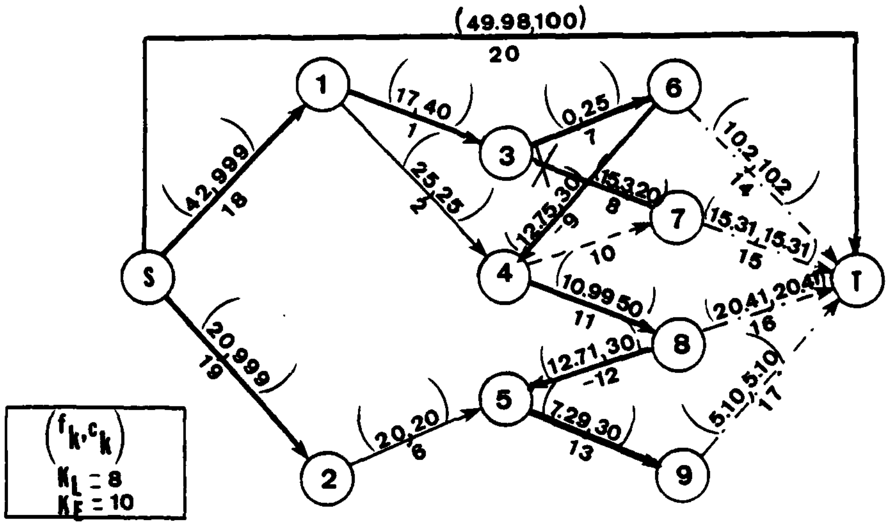
(b)
Example Application of PARARC Algorithm
Figure 6.4(cont)



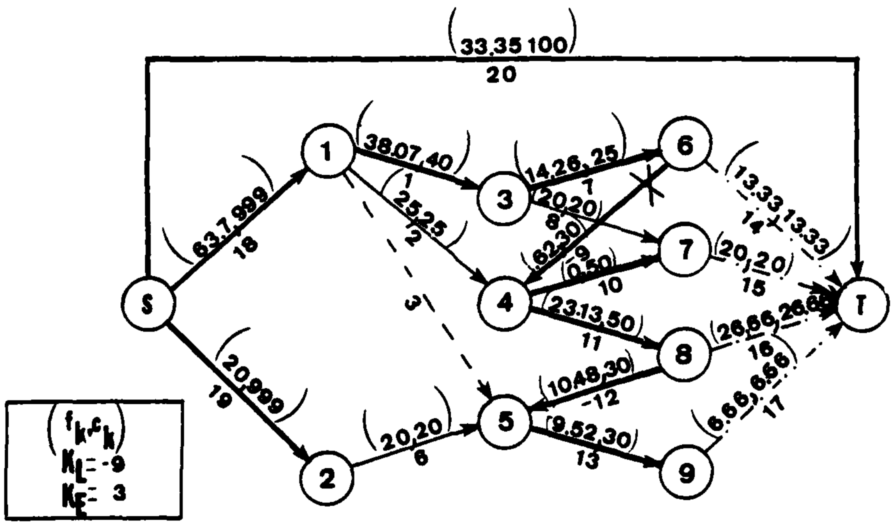
(c)
Example Application of PARARC Algorithm
Figure 6.4(cont)



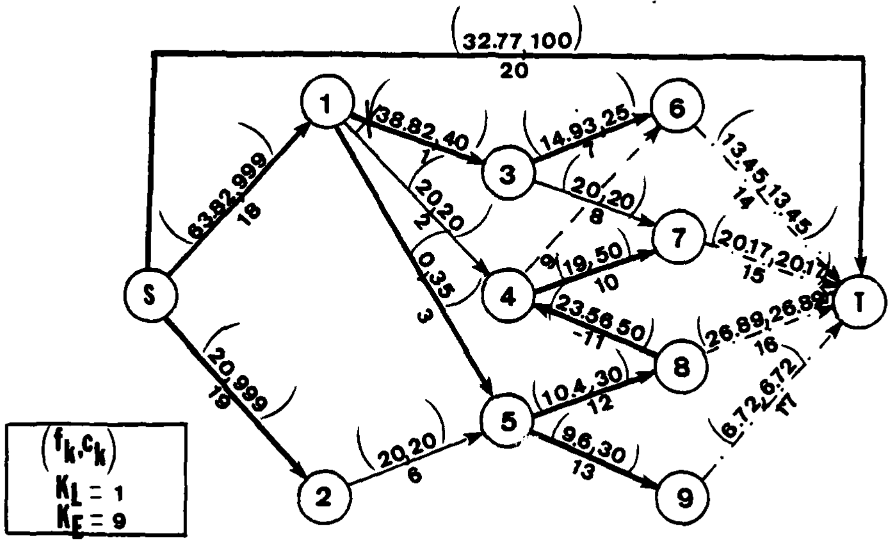
(d)
Example Application of PARARC Algorithm
Figure 6.4(cont)



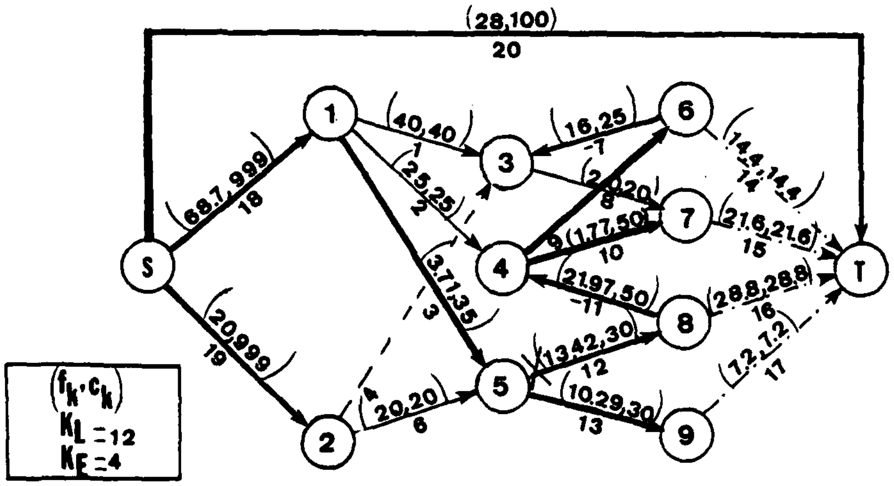
(e)
Example Application of PARARC Algorithm
Figure 6.4(cont)



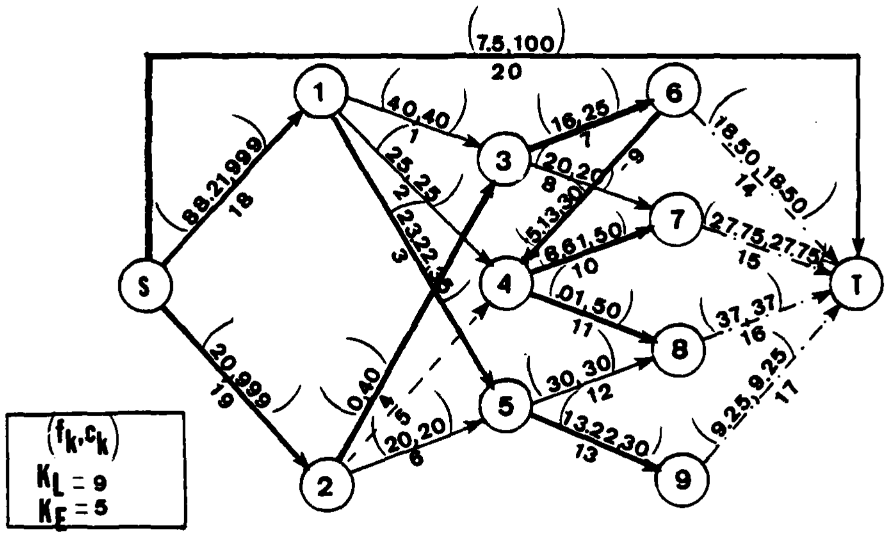
(f)
Example Application of PARARC Algorithm
Figure 6.4(cont)



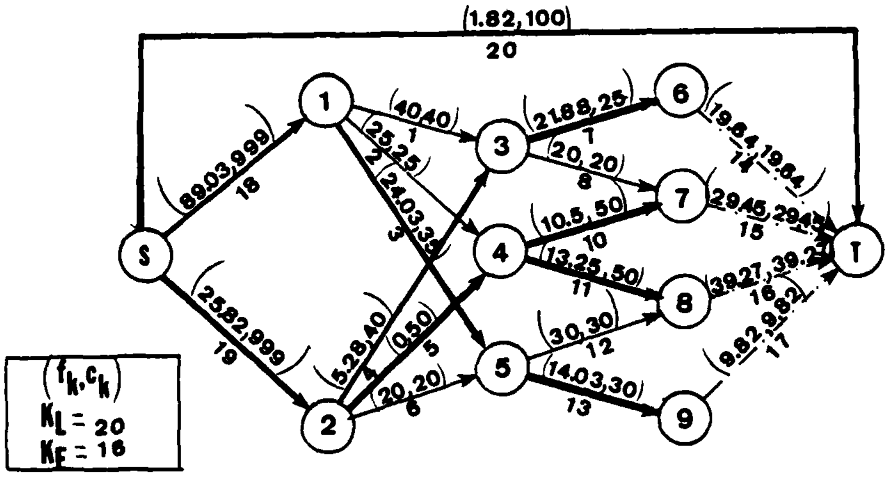
(g)
Example Application of PARARC Algorithm
Figure 6.4(cont)



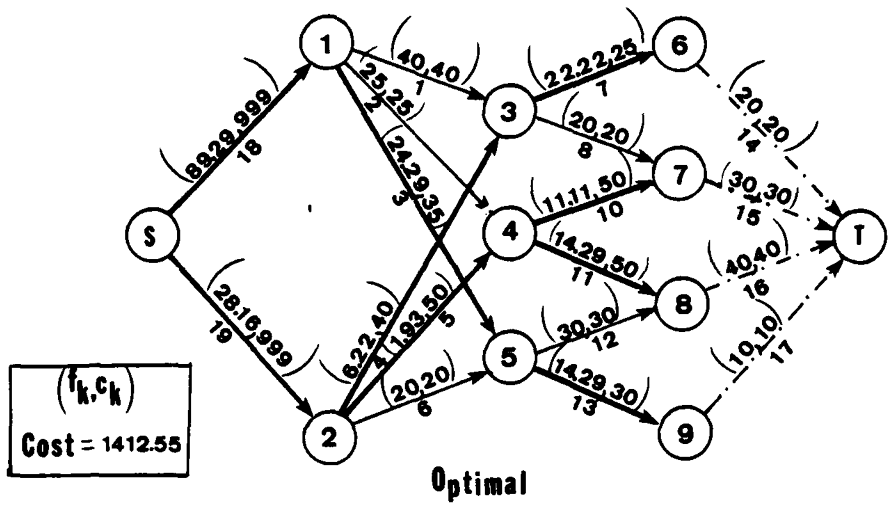
(h)
Example Application of PARARC Algorithm
Figure 6.4(cont)



(i)
Example Application of PARARC Algorithm
Figure 6.4(cont)



(j)
Example Application of PARARC Algorithm
Figure 6.4(cont)



(k)
Example Application of PARARC Algorithm
Figure 6.4(cont)

Figure 6.4(k) shows the optimal solution as the flow on the artificial arc is reduced to zero.

6.3 Parametric Node Incremental Flow Algorithm

Given a network with arcs having capacity, cost and gain parameters, with source and destination nodes defined, and required quantities of flow specified for each source node and destination node, the problem is to determine the arc flows with the minimum total cost. The flows must satisfy the conservation of flow except at the super-source. While the flows into each destination must equal the required values, the super-source may supply any amount of flow. The problem may be stated

$$\text{Min} \quad \sum_{k=1}^{m} h_k f_k \qquad (6.8)$$

$$\text{S.T.} \quad -\sum_{k \in M_{Ti}} a_k f_k + \sum_{k \in M_{Oi}} f_k = b_i \quad \text{for } i \in N-s \qquad (6.9)$$

$$0 \leq f_k \leq c_k \ . \qquad (6.10)$$

The parametric node incremental algorithm is a dual-node-infeasible approach applied to a specialized network. The specialized network has a single source, but perhaps multiple destination nodes

We can define an intermediate primal problem as follows:

$$\text{Min} \quad \sum_{k=1}^{m} h_k f_k \tag{6.11}$$

$$\text{S.T.} \quad -\sum_{k \in M_{Ti}} a_k f_k + \sum_{k \in M_{Oi}} f_k = b_i^{\,1} \quad \text{for } i \in N-s \tag{6.12}$$

$$0 \le f_k \le c_k \qquad k \in M \tag{6.13}$$

where $b^1_i \le b_i$ for $i \in N - 1$. An intermediate optimum is a flow F_1 and node potentials π_1 that are an optimum solution to the above primal problem and its associated dual.

The general approach begins with an intermediate optimum F_0 for some value of output flow at each of the sinks. Next a shortest path network is constructed, $D_s^{\,0}$, that determines for each node the minimum cost per unit of additional flow to the sink nodes and the path over which that flow may be obtained. The output flow at each destination node is increased parametrically in $D_s^{\,0}$ along the minimum cost trails defined to all the destinations until an arc becomes capacitated. This flow augments F_0 to F_1. F_1 is also an intermediate optimum. A new augmenting network is constructed, $D_s^{\,1}$, and the output flow is again augmented to obtain F_2. This process continues iteratively until the desired output flows at the destination nodes are met.

Throughout the algorithm because the values of π_i are determined by a shortest path algorithm, the solution will remain dual feasible. The solution is always basic as at most n-1 arcs will have flows strictly between their bounds, and these arcs will form a basis network. The solution will satisfy conservation of flow and complementary slackness. The one requirement that is violated until the final iteration is the requirement for the flow at each of the destination nodes to be satisfied. These flows start at zero and iteratively increase until they reach their required values. At this point the algorithm stops with the optimum solution. The algorithm has the following basic steps:

1. Find an initial least-cost spanning tree.
2. If all sink external flows are met, stop with the optimal solution.
3. For each sink node, find the flow augmenting trail. Parametrically increase the flow on all trails. Determine the maximum allowable flow increment and determine the arc to leave the network.
4. Find an arc to enter the network. If none exists, stop. There is no feasible solution.
5. Change the basis by deleting the leaving arc and inserting the entering arc. Modify the dual variables to satisfy complementary slackness. Return to step 2.

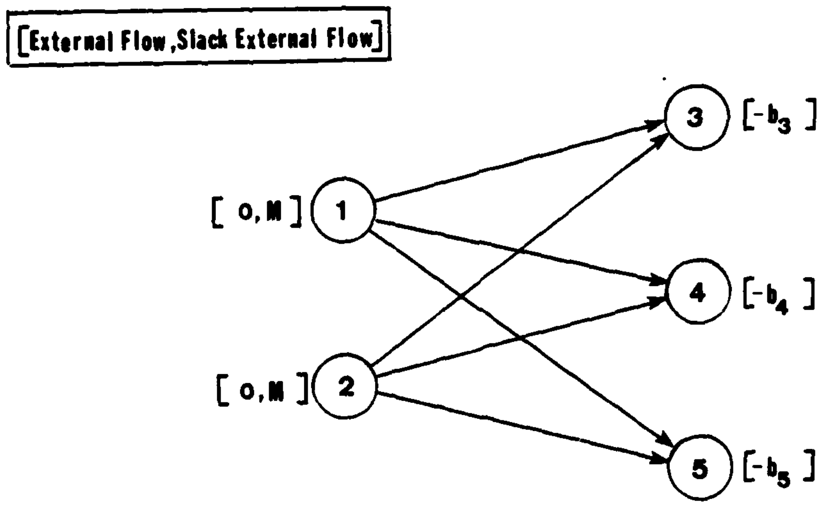
Again before stating the algorithm in complete detail, each step of the outline will be discussed.

6.3.1 Obtaining the Initial Basis Tree

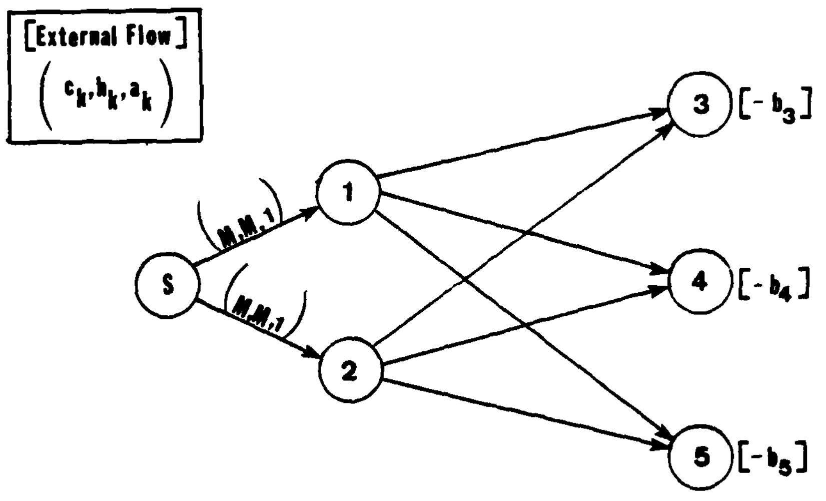
In order to apply the concepts developed above and the ones presented in Chapter 5, we need to modify the network as presented. Consider the sample network shown in Figure 6.5. We create a super-source and arcs from the super-source to each source node with parameters:

$$c_k = b_{si}$$
$$h_k = 0$$
$$a_k = 1 .$$

This transformation is accomplished by the subroutines READNDE and ORIGSG. The revised network is presented in Figure 6.6. Once the transformed network is obtained, we use a generalized Dijstra's algorithm as documented in [46,page 283] to obtain in initial basic solution. We then use the subroutine TREINT [46,page 121] to initialize the parameters of the basis tree. Since a generalized network problem may have arc gains greater than one or have negative



Sample PARANDE Problem
Figure 6.5



The Augmented Network
Figure 6.6

arc costs, Dijkstra's algorithm may not provide an optimal shortest path tree. Therefore, we now use the algorithm PSHRTG [46,page 285] to find the initial least-cost spanning tree. At this point we have an initial optimal (all $d_k \geq 0$), basic, node infeasible solution to the original problem.

6.3.2 Determining the Arc to Leave the Basis

As in PSENG, we now use the PATH subroutine to determine the flow augmenting paths. In this case, however, the candidate list contains node numbers, and the node parametric parameters are the negative of the respective destination node external flows. For each destination node we find the flow augmenting trail. This will create a list of arcs (LISA), a list of nodes (LISN), and a gamma list (G). Using these lists, the subroutine MFLOWG [46,page 266] will determine the maximum flow change or reference number on these trails and the limiting arc k_L.

At each iteration we keep a record of the cumulative reference number. This is merely the sum of the reference numbers to this iteration. It is possible that the reference number found for a given iteration, i, when added to the cumulative reference number from the previous iteration i-1, will be greater than 1. This means a reference number of (1 - cumulative reference number) will cause the unsatisfied external flows to be satisfied exactly. At this point we would augment the flows and terminate with the optimal solution.

6.3.3 Determining the Arc to Enter the Basis

We use the subroutine FINDNDE to determine the entering arc k_E. If no arc can be found to enter the basis, no feasible solution exists to the proposed problem.

6.3.4 Changing the Basis

Updating the basis tree representation and the dual variables is accomplished by the subroutine PIVOT1G.

6.3.5 The Complete Algorithm

We can now give a complete statement of the parametric node incremental algorithm.

1. Initialize all flows to zero.

2. Use a variant of Dijstra's shortest path algorithm to find the initial least-cost spanning tree. Use the arcs in the spanning tree to define a set of node labels π_i such that $\pi_s = 0$ and $\pi_j = (\pi_i + h_k) / a_k$ for all arcs $k(i,j) \varepsilon M_T$.

3. Use a triple labeling scheme to label and store the least cost spanning tree.

4. If the cumulative reference number (CUMRNO) equals 1.0, stop with the optimal solution.

5. For each demand node find the flow augmenting trail. Determine the maximum reference number, MF, on these trails. If the maximum MF is such that the CUMRNO + MF > 1, set MF = 1 - CUMRNO and terminate the algorithm after this step. Determine the arc on these trails which is at capacity and designate it the leaving arc k_L. Augment the flows.

6. Delete the arc k_L from the basis tree, partitioning the network into two subtrees defined by the set N_1 and the set N_2. Search for an admissible arc that originates or terminates in N_1 and connects to the set N_2, or an admissible arc that originates and terminates in N_2. If no such arc exists, stop; there is no feasible solution to the problem. Otherwise select the arc with the minimum d_k and designate it as the entering arc k_E.

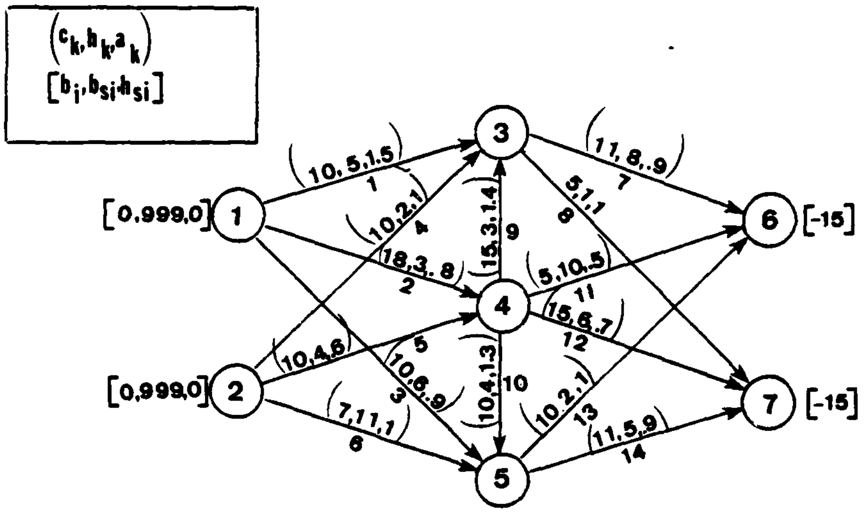
7. Perform a basis change which determines the new least-cost spanning tree. The basis change is accomplished by deleting k_L and adding k_E. Update the dual variables and return to step 4.

The algorithm PARANDE implements these steps. The flow charts for the FORTRAN coding of the algorithm can be found in Appendix 2.

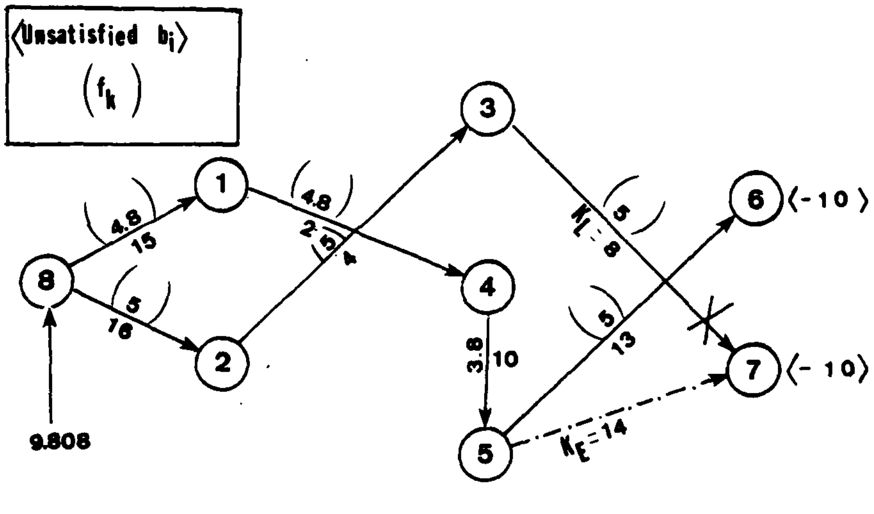
Figure 6.7 presents an example application of the algorithm PARANDE. Figure 6.7(a) shows the original problem data. In each of the Figures 6.7(b) through 6.7(h), the entering arc is indicated by a dashed-dot line, the leaving arc by an X, the basic arcs by solid lines and the nonbasic arcs at capacity by dashed lines. The unsatisfied external flows are also shown at each step. Figures 6.7(b) through 6.7(g) show the intermediate steps in the solution process. In Figure 6.7(g) no leaving arc is shown, as the maximum flow augmentation found is greater than the maximum flow augmentation necessary to satisfy the external flows. The algorithm terminates at this point and the optimal solution is shown in Figure 6.7(h).

6.4 Computational Analysis of PARARC and PARANDE

To demonstrate the feasibility of using parametric analysis for incremental flow algorithms, PARARC and PARANDE were tested against their immediate predecessor INCREMG [46,page 314]. INCREMG, in contrast to the parametric approach, only augments flow along a single flow augmenting path. In the parametric approach flow is increased in multiple flow augmenting paths. The computations were conducted to determine if these new algorithms would be more efficient than INCREMG. The computational times were also compared against two primal codes, NETG (38) and PGAINS (46). We should note that both PGAINS and INCREMG are modular and pedagogical in design,



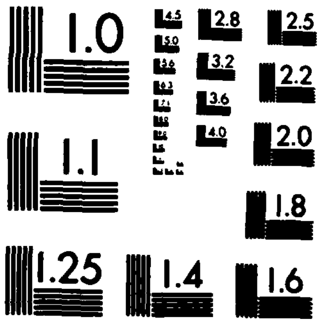
(a)
Example Application of PARANDE Algorithm
Figure 6.7

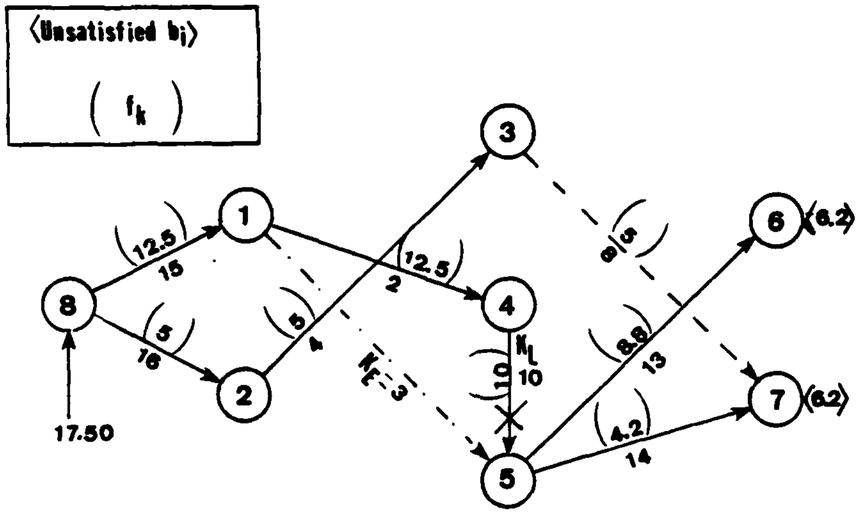


(b)
Example Application of PARANDE Algorithm
Figure 6.7(cont)

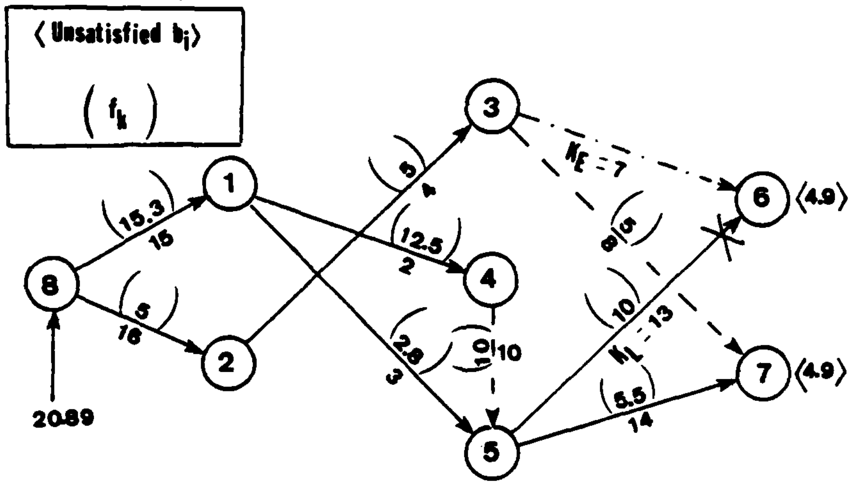
AD-A145 558 PARAMETRIC ANALYSIS FOR GENERALIZED NETWORK FLOW PROBLEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH M E BAUM MAY 84 AFIT/CI/NR-84-38D 3/3

UNCLASSIFIED F/G 12/1 NL

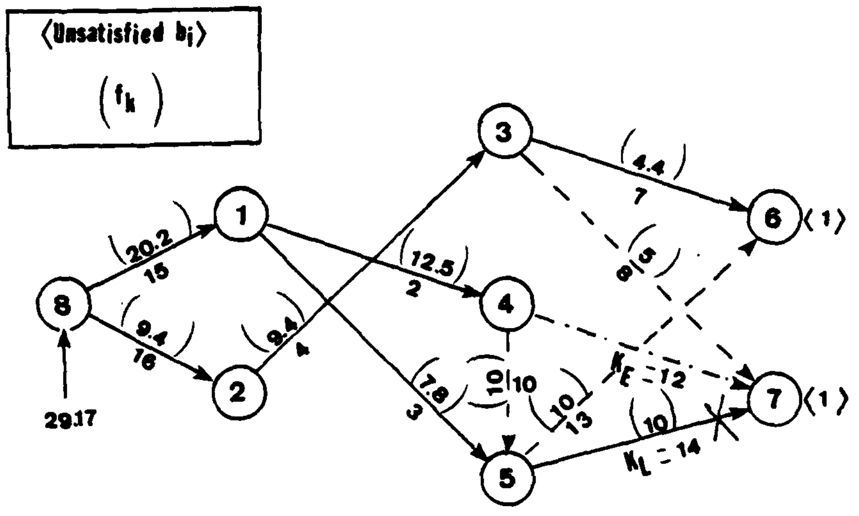

1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6



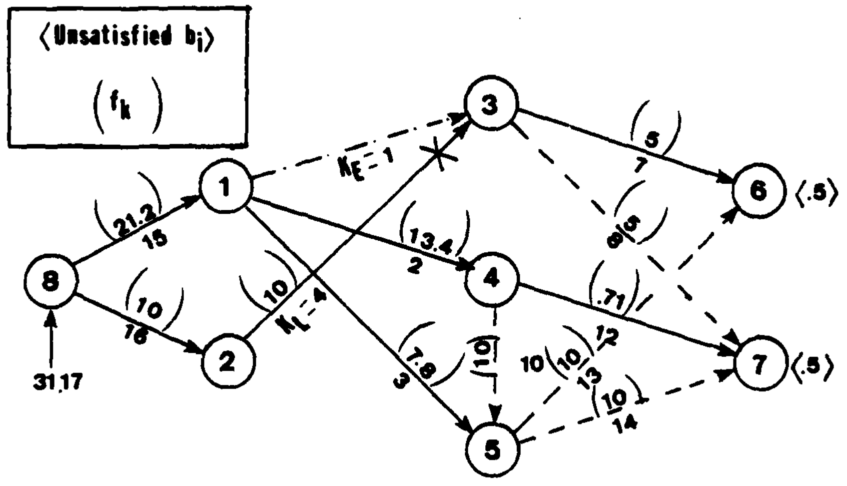
(c)
Example Application of PARANDE Algorithm
Figure 6.7(cont)



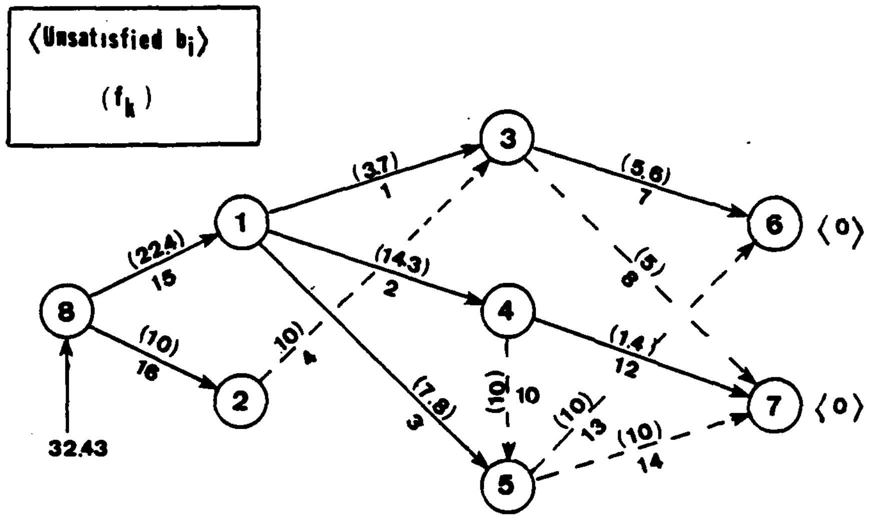
(d)
Example Application of PARANDE Algorithm
Figure 6.7(cont)



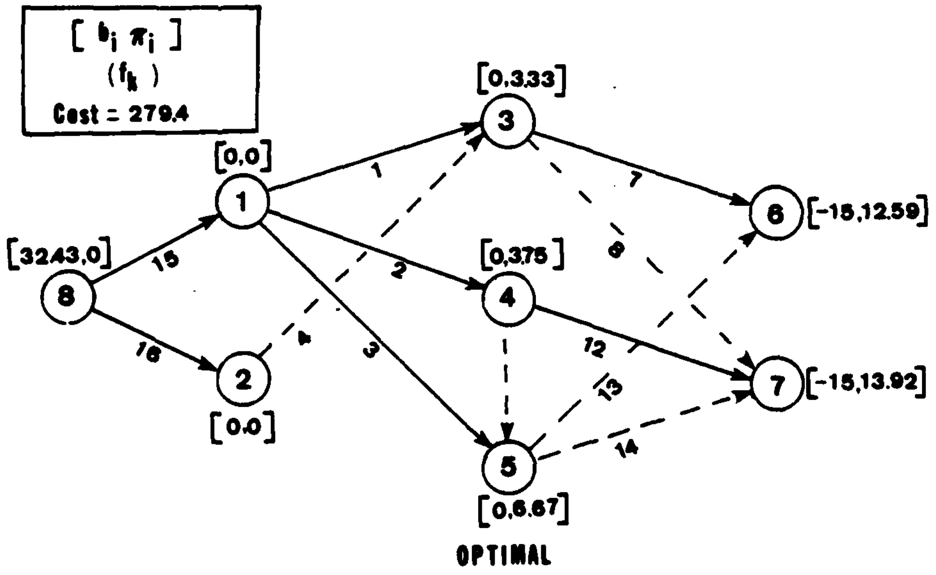
(e)
Example Applicatiion of PARANDE Algorithm
Figure 6.7(cont)



(f)
Example Application of PARANDE Algorithm
Figure 6.7(cont)



(h)
Example Application of PARANDE Algorithm
Figure 6.7(cont)



(g)
Example Application of PARANDE Algorithm
Figure 6.7(cont)

and this may influence the computational times recoreded.

A set of 20 test problems (see Table 6.1) included in the computational analysis were generated by a random-network generator NETGNG. While NETGNG has no specific reference, it is in use by the Center for Cybernetic Studies at The University of Texas at Austin. NETGNG's immediate predecessor for pure networks is NETGEN (53). NETGEN, as part of its procedure, builds a skeleton network connecting sources to sinks. Capacities are assigned to all skeleton branches to assure the existance of a feasible solution which uses all supplies and meets all demands. The sum of supplies equals the demand. Nonskeleton branches are then randomly generated. The primary modification to this procedure, used by NETGNG, for networks with gains is also to assign nonskeleton branches a gain parameter randomly selected from a specific range. The skeleton branches are given unity gains. This guarantees the existance of a feasible solution.

The problem set includes 10 transportation and 10 transshipment networks. The number of nodes varies from 60 to 100 and the number of arcs from 100 to 600. The cost on the arcs ranges between 1 and 100. All problems have 100% capacitated arcs, with minimum lower bound of 5 and maximum upper bound of 10. Arc gains range between 0.5 and 1.5. All problems were generated by using a seed of 12345678 and 80% of the arcs were assigned the highest

Problem Number	Number of Nodes	Number of Sources	Number of Sinks	Number of Arcs	Cost Range	Total Supply	Capacity	Bound Min	Bound Max	Gains Min	Gains Max
1	100	5	95	100	1-100	10,000	100	5	10	0.5	1.5
2	100	5	95	250	1-100	10,000	100	5	10	0.5	1.5
3	100	5	95	300	1-100	10,000	100	5	10	0.5	1.5
4	100	10	90	300	1-100	10,000	100	5	10	0.5	1.5
5	100	10	90	350	1-100	10,000	100	5	10	0.5	1.5
6	100	10	90	400	1-100	10,000	100	5	10	0.5	1.5
7	100	10	90	500	1-100	10,000	100	5	10	0.5	1.5
8	60	10	50	100	1-100	10,000	100	5	10	0.5	1.5
9	60	10	50	300	1-100	10,000	100	5	10	0.5	1.5
10	60	10	50	400	1-100	10,000	100	5	10	0.5	1.5
11	75	5	50	300	1-100	10,000	100	5	10	0.5	1.5
12	75	5	50	400	1-100	10,000	100	5	10	0.5	1.5
13	75	5	50	500	1-100	10,000	100	5	10	0.5	1.5
14	75	5	50	600	1-100	10,000	100	5	10	0.5	1.5
15	85	7	55	300	1-100	10,000	100	5	10	0.5	1.5
16	85	7	55	400	1-100	10,000	100	5	10	0.5	1.5
17	85	7	55	600	1-100	10,000	100	5	10	0.5	1.5
18	95	10	60	400	1-100	10,000	100	5	10	0.5	1.5
19	95	10	60	500	1-100	10,000	100	5	10	0.5	1.5
20	95	10	60	600	1-100	10,000	100	5	10	0.5	1.5

Computational Testing Problem Data
Table 6.1

possible cost.

The five algorithms were run on a Cyber 170/750A computer at The University of Texas at Austin. All codes were tested on FTN4 (OPT = 2) compiler. Tables 6.2 and 6.3 display the results from NETG, PGAINS, INCREMG, PARARC, and PARANDE. All solution times include only the time to optimize the problem excluding times to input the data and output the results.

It can be concluded based on the limited sample results shown in Tables 6.2 and 6.3 that both primal codes completely dominated the dual-incremental codes by between two to five times. However, PARANDE is more efficient than either PARARC or INCREMG. For the 10 transportation problems PARANDE is 1.41 times faster than PARARC and 1.67 times faster than INCREMG. For the 10 transshipment problems PARANDE is 1.27 times faster than PARARC and 1.67 times faster than INCREMG. Overall PARANDE is 1.31 times faster than PARARC and 1.67 times faster than INCREMG.

It can also be concluded that PARARC is more efficient than INCREMG. For the transportation problems PARARC is 1.18 times faster than INCREMG and for the transshipment problems PARARC is 1.31 times faster than INCREMG. Overall PARARC is 1.27 times faster than INCREMG.

The major objective of this computational study was to demonstrate the feasibility of applying parametric analysis to

	PRIMAL METHODS		DUAL METHODS			
Problem Number	NETG	PGAINS	INCREMG	PARARC	PARANDE	Objective Function
1	0.316	0.968	1.032	0.958	0.622	844765.0
2	0.960	1.425	1.966	2.087	1.383	743281.6
3	0.942	1.587	3.863	3.965	2.697	801090.7
4	0.657	1.761	4.155	3.931	2.635	840359.3
5	1.043	1.811	5.157	4.717	4.125	746985.6
6	0.632	2.029	6.183	5.542	3.745	812475.3
7	1.260	2.594	10.682	8.066	5.487	757218.0
8	0.131	0.564	0.670	0.687	0.487	831556.8
9	0.565	1.138	3.881	3.151	2.289	715154.7
10	0.693	1.507	4.775	2.689	1.866	743944.5

Solution Times for the Transportation Problems
Table 6.2

	PRIMAL METHODS		DUAL METHODS			
Problem Number	NETG	PGAINS	INCREMG	PARARC	PARANDE	Objective Function
11	0.453	1.146	4.738	4.664	3.666	2357360.3
12	0.828	1.449	7.096	5.408	4.276	2085017.7
13	0.936	1.812	10.432	7.243	5.662	2203213.5
14	1.516	2.310	18.143	13.394	10.959	2082679.7
15	0.915	1.296	5.068	5.471	4.223	2230957.7
16	0.913	1.587	8.100	7.072	5.655	2473342.4
17	1.143	2.929	15.195	9.785	7.746	2117899.7
18	0.969	1.886	8.440	6.375	4.696	2055368.9
19	0.986	2.139	12.480	10.116	7.698	1951960.6
20	1.616	2.925	21.410	15.285	12.053	2169286.6

Solution Times for the Transshipment Problems
Table 6.3

dual-incremental methods. While the codes tested were not as efficient as primal codes, they were considerably more efficient than the previous generation of dual-incremental codes. This does indicate an area of promising research which will be discussed in Chapter 7.

6.5 Other Applications of Parametric Analysis

In this section we will describe four additional problem classes where parametric sensitivity analysis for generalized networks might be applied, with what we feel would be with a great deal of success.

The first is the cash management problem (9) which is concerned with optimally financing net cash outflows and investing net cash inflows of a firm while simultaneously determining payment schedules for incurred liabilities. This could be formulated as a multi-period transshipment model with m "warehouses" corresponding to different sources of cash (cash sales, short-term securities, lines of credit, etc.) and the n "markets" corresponding to different uses of cash (payments on accounts, notes payable, etc.). Cash could be both a source and a use, thus allowing transshipments across periods. Yields on securities, interest rates on loans, and discounts constitute unit costs of the problem. While the model takes into account payment schedules, financing and security transactions, it ignores the important problem of determining the minimum balance. Cash deposits in excess of some absolute minimum cash balance requirement prove valuable since they improve the firm's credit rating and the bank's goodwill. But deposits also mean lost revenue from alternative investments in securities. Thus the manager would like to know the effect on the optimum total cost from maintaining different levels of cash balance, so as to arrive at a subjective decision regarding its magnitude. This can be easily accomplished by parametric analysis of the cash balance (external flows).

A second major area is the multi-period multi-region capacity expansion problem. Balachandran, et al. (9), describe this problem. A product can be manufactured in m regions and is required by n markets for each of the periods $K = \{1,2,...,T\}$. We know that the initial demand and the demand in a market can be satisfied by production and shipment from any region but must be met exactly during each time period (no backlogging or inventorying). Balachandran notes that each period is a transportation problem, and the problem for period t differs from the problem in period t-1 by only the right-hand side constraints. Thus we can iteratively solve the problem. Starting in period 1 and using the solution from period 1, we use parametric analysis to get the solution for period

2. We can continue in a like manner to get the solution for period T.

A third major area would be in the area of branch-and-bound algorithms. It often happens that the subproblems are generalized network flow problems. Consequently, as we move along the branch-and-bound tree, we may obtain the optimal solution to the (k+1)st subproblem by post-optimization of the kth subproblem, rather than inefficiently having to solve the (k+1)st problem from the start. Some of the potential areas within branch-and-bound might deal with optimal facility location, assignment of sources to users, lock-box decision models, or the one-machine job scheduling problem.

A final area where this method might make solution methods more efficient is in the area of stochastic network flow problems with recourse (45). In this problem one really has a master problem and a series of subproblems. Each of the subproblems is a network where the right-hand side is being varied at each iteration. It seems that the parametric sensitivity algorithm would be of value to solve these types of problems, as the solution for iteration k could be used to start iteration k+1 by just varying the capacity constraints.

While the four example application areas do not exhaust the potential areas of applications for the parametric analysis method for generalized networks, we hope that these examples may spur further applications along these lines.

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Summary

In a deterministic generalized network after the initial model is constructed, the operations research scientist must still gather the data to completely fill in the parameters of the model. The uncertainty and inaccuracy of the initial data collection is well documented. When the operations research scientist has presented these results to management, the scientist has many times met with failure because management challenges the original problem data. The real strength of linear programming has been that the operations research scientist has been able to use sensitivity analysis or parametric programming to perturb the initial problem data to answer management's "what if" questions.

A logical extension of parametric programming of linear programs would be to apply these well known concepts to a generalized network format. The goal of this research was to develop the individual algorithms to accomplish this both by parametric analysis for a capacity vector, as well as parametric analysis of the right-hand side or external flow vector. In order to accomplish this goal, an algorithm to determine the flow augmenting trails (TRAIL2), an algorithm to determine the maximum flow augmentation possible (MFLOG2), an algorithm to determine the entering arc (FIND2G), and finally an algorithm to change the basis (PIVOT1G) were developed.

A computer package was developed which integrated these algorithms with a report generator. The report allows the manager to see the optimal flows at each iteration and examine how the total cost varies with each basis change.

An outgrowth of this research has been the development of two new variants of a dual-incremental flow approach to the generalized network flow problem.

7.2 Contributions

As described in Chapter 2, much work historically has been spent on parametric programming for linear programs. This is in contrast to a lack of significant work in the area of parametric programming for the generalized network flow problem. There are two major contributions of this research. First, the methods developed in this study have not been seen before in the literature and they are a significant tool for the practical operations research analyst. The methods allow the analyst to perturb an initial uncertain data set to analyze how these perturbations will effect recommendations to management. The computer code developed allows the analyst to present a graphical report on how the objective function varies as the parameters are varied. This leads to an easy visualization of how much to vary the parameters or how much the parameters may be varied before an adverse effect is likely to happen to the company.

While the first contribution is of great practical value, the second outgrowth of this research has been the development of two new dual-incremental codes. While these codes are not numerically superior to primal methods, they do extend the theory of parametric analysis to solve generalized network flow problems. Before this research, parametric analysis as related to operator theory was used only to solve the generalized transportation problem (5,6,7,8).

7.3 Recommendations for Further Research

There are three areas which could lead to further research. The first logical extention would be to develop parametric programming codes for changes in arc costs or gain parameters. This would allow the operations research analyst to vary all the parameters of the network to do a complete analysis of the proposed model.

The second area would be to continue further testing to make PARARC and PARANDE computationally more efficient. In (67) Schmitt developed a pure dual-incremental code which is relatively competitive with known primal codes. This leads to speculation that dual-incremental codes for generalized networks might also be competitive if they were modified appropriately. The principle time consumer for a dual method is the search for the arc to enter the basis. This usually requires a search in each iteration proportional to the number of arcs. In a private communication (58) Matsumoto suggests that one only need search through those admissible arcs which originate or terminate in the N_2 set. Another potential way to reduce computational time is to implement PARARC and PARANDE with the augmented threaded index method. Glover, Klingman, and Stutz report a 10% reduction in computation time using preorder traversal lists for the basis representation rather than the triple label method.

A third way to improve PARARC and PARANDE would be the way the initial basis tree is found. While the basis initialization only consumes 2-3% of the total optimization time, it is possible that the algorithm THRESH presented by Glover, et al. (39) may be able to reduce this even further. A fourth possible way to save time is to eliminate the use of mirror arcs in a way similar to

Schmitt's work (67). The comparison to check if arc k is greater than zero is done numerous times throughout the complete algorithm. Without counting the number of times this logical check is done after the computations with do loops are applied, the author has roughly counted over 40 individual logical comparisons of $k > 0$. A fifth possible way to lower the optimization time is to continue to increment the flow on those arcs on the tree until all paths from the source to the destinations are capacitated. Then by using a modified PSHRTG a new basis tree could be found. This could potentially cut down on the number of dual iterations, thus greatly reducing computational time.

A final area of research interest would be to apply the parametric methods to multicommodity flow problems. There has been much recent interest in this area (16,34,35,51,60,65). It seems possible that a two commodity flow problem in the form:

$$\begin{aligned} \text{min} \quad & H_y F_y + H_x F_x \\ \text{S.T.} \quad & AF_y = b_y \\ & AF_x = b_x \\ & 0 \le F_x \le GF_y \\ & F_y \ge 0 \end{aligned}$$

could be solved by dividing this problem into a master problem and a subproblem. In the subproblem the concepts of parametric analysis could be used to modify the right-hand sides and then pass the required data back to the master problem.

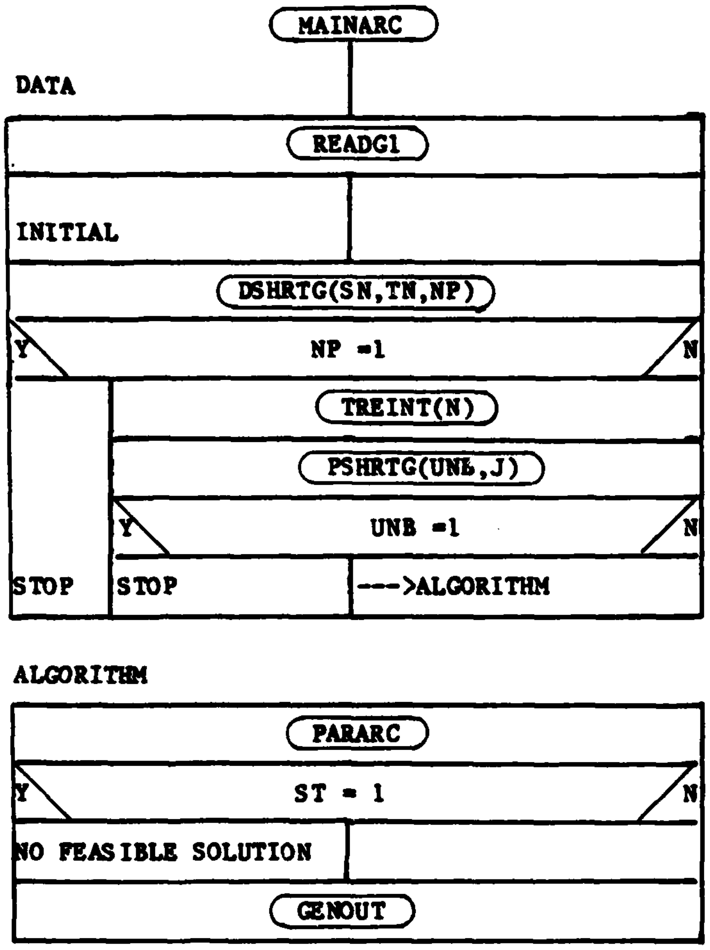
APPENDIX 1

PURPOSE: To read data, initialize the basis, and call parametric arc incremental algorithm to generalized network problem.

1.(DATA) Read problem data from input source.

2.(INITIAL) Use DSHRTG to find the shortest path from slack node to super-sink node. If no path exists (NP =1), stop. Otherwise use TREINT to construct pointer representation of tree. Call PSHRTG to find shortest path for generalized net work. If UNB =1, an unbounded solution exists. Otherwise go to 3.

3.(ALGORITHM) Call pararc to solve problem. If ST = 1 there is no feasible solution. GENOUT prints solution results.



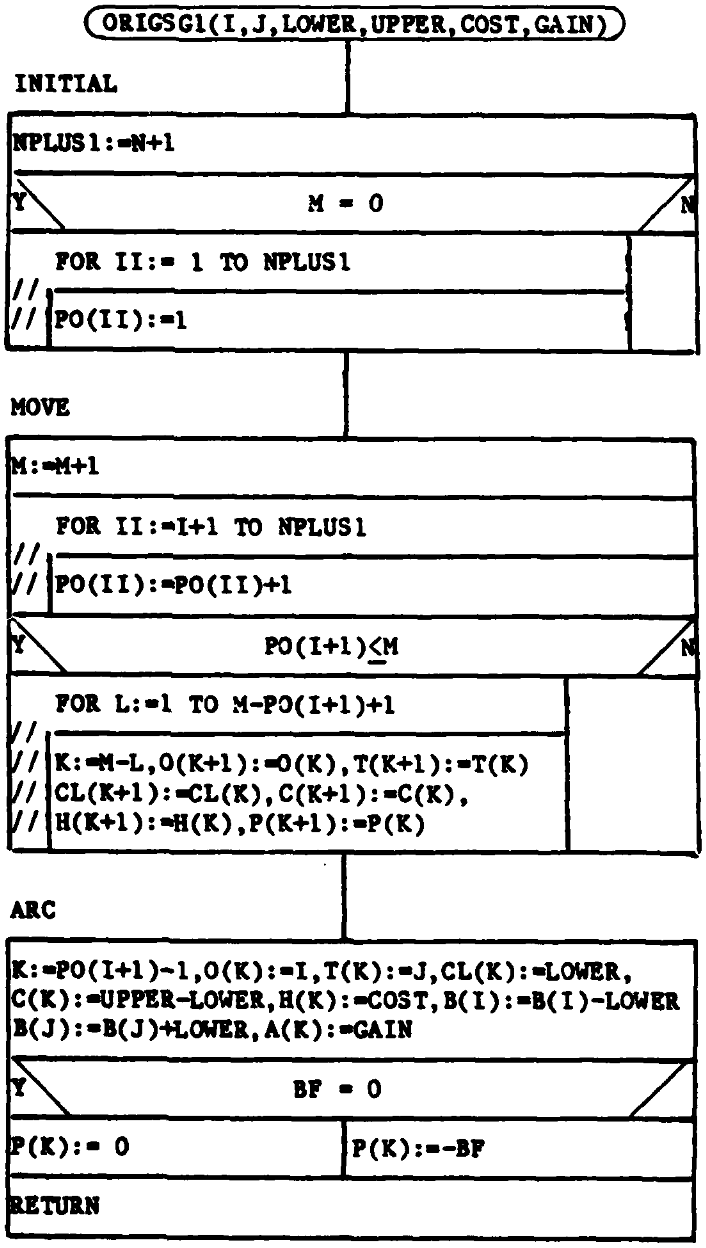
Main Program MAINARC

PURPOSE: To accept arc data item and store it in an arc list ordered by ascending origin node.

1.(INITIAL) If first call to ORIGSG1, set all node pointers to 1. Otherwise go to 2.

2.(MOVE) Increase M by one. Increase all node pointers greater than I by one. Move all arcs above the new entry one index higher on list.

3.(ARC) Insert arc in last position alloted to node I. Modify the upper bounds of the arcs and fixed external flows to account for the arc lower bound.



```
ORIGSG1(I,J,LOWER,UPPER,COST,GAIN)
INITIAL
    NPLUS1:=N+1
    M = 0 ?
    Y:  FOR II:= 1 TO NPLUS1
            PO(II):=1
    N:
MOVE
    M:=M+1
    FOR II:=I+1 TO NPLUS1
        PO(II):=PO(II)+1
    PO(I+1)≤M ?
    Y:  FOR L:=1 TO M-PO(I+1)+1
            K:=M-L,O(K+1):=O(K),T(K+1):=T(K)
            CL(K+1):=CL(K),C(K+1):=C(K),
            H(K+1):=H(K),P(K+1):=P(K)
    N:
ARC
    K:=PO(I+1)-1,O(K):=I,T(K):=J,CL(K):=LOWER,
    C(K):=UPPER-LOWER,H(K):=COST,B(I):=B(I)-LOWER
    B(J):=B(J)+LOWER,A(K):=GAIN
    BF = 0 ?
    Y:  P(K):= 0
    N:  P(K):=-BF
    RETURN
```

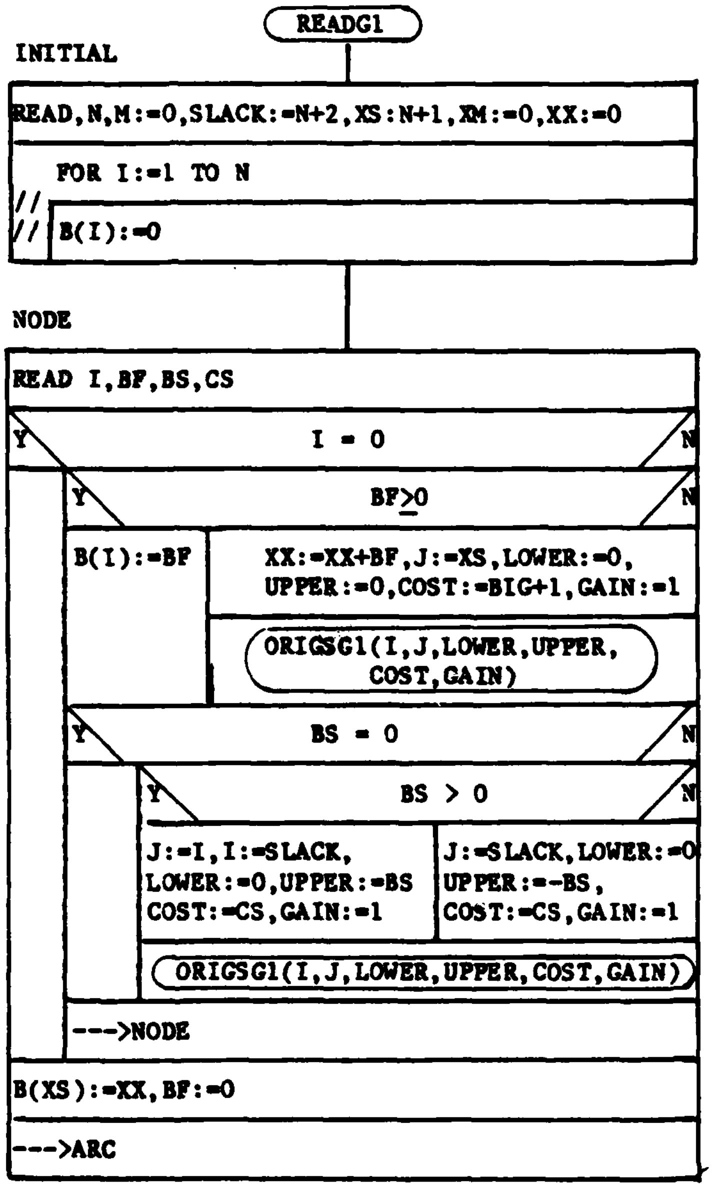
Subroutine ORIGSG1

PURPOSE: To read and store node and arc data for the minimum cost flow problem.

1.(INITIAL) Initialize the number of arcs to zero. Create super-slack node and super-sink node (xs). Set all external flows to zero.

2.(NODE) Read a node data item.If fixed external flow is positive,put fixed flow in storage.Otherwise, add up the cumulative amount of demands and create arc from demand node to xs. If slack external flow is zero go to 2. Otherwise create a slack arc and store data in correct position in arc lists. After all node data has been read (I=0). Set super-sink external flow to cumulative demands. Go to 3.

3.(ARC) Read an arc data item. If I=0,



Subroutine READG1

go to step 4 as all arc data has been read. Else store the arc data in the correct position in the arc lists. Repeat step 3.

4.(ARTIFICIAL) Create an artificial arc from the super sink to each supply node.Place arc in correct position in arc lists. Create candidate arc(XK) and number of arcs in list(XM). Set the counter for the sink node (TN) and slack node(SN).Go to 5.

5.(EXT) Place each arc's data in the correct position in the terminal list.

```
ARC
READ I,J,LOWER,UPPER,COST,GAIN
I = 0
  Y: --->ARTIFICIAL
  N: ORIGSG1(I,J,LOWER,UPPER,COST,GAIN)
     --->ARC

ARTIFICIAL
FOR IG:=1,N-1
    B(IG) < 0
      Y: I:=SLACK,J:=IG
         LOWER:=0,COST:=BIG
         UPPER:=-B(IG),GAIN:=1
      N: I:=IG,J:=SLACK
         LOWER:=0,COST:=BIG
         UPPER:=B(IG),GAIN:=1
    ORIGSG1(I,J,LOWER,UPPER,COST,GAIN)
FOR I:=1 TO M
    P(I) = 0
      Y:
      N: XM:=XM+1,XK(XM:=I
SN:=SLACK,TN:=0

EXT
LM:=M,M:=0
FOR K:=1 TO LM
    J:=T(K),M:=M+1
    TERMS(K,J)
RETURN
```

Subroutine READG1(cont)

PURPOSE: To implement the parametric arc flow augmentation approach for a generalized network.

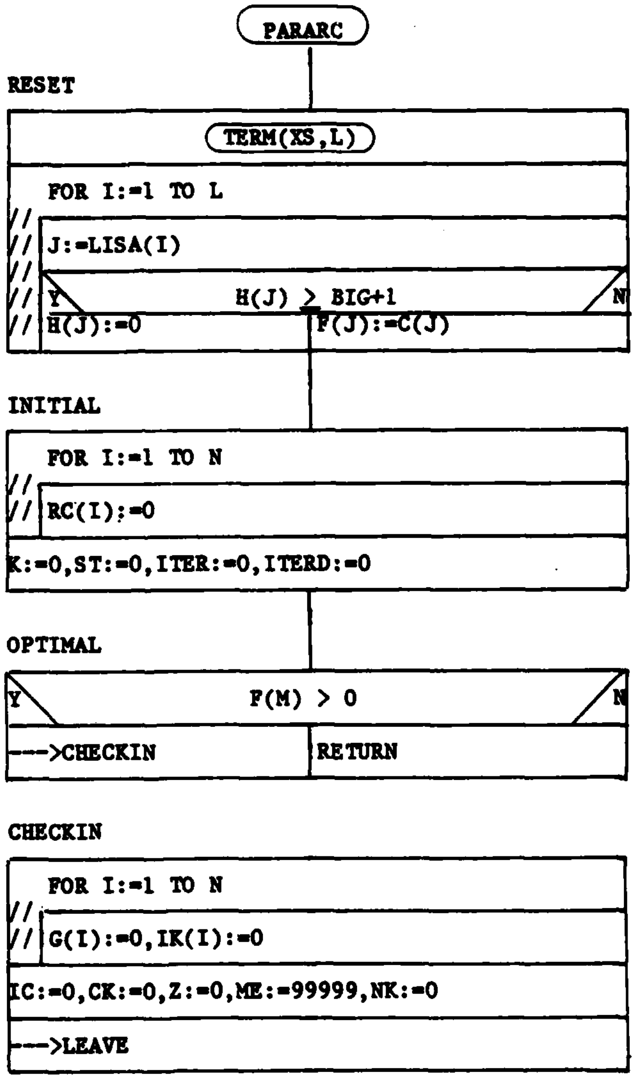
1.(RESET) Set the costs on all arcs terminating at the super-sink node to zero, except the artificial arc.

2.(INITIAL) Initialize all parameters to zero.

3.(OPTIMAL) If the flow on the artificial arc equals 0, return with optimal solution. Otherwise go to 4.

4.(CHECKIN) Initialize the ϒ's to zero. Initialize arc,node, and cycle counters to zero.

5.(LEAVE) For each arc on the candidate arc list, determine if the arc is nonbasic. If it is and its parametric parameter is less than zero, calculate a refer-



```
PARARC
RESET
    TERM(XS,L)
    FOR I:=1 TO L
        J:=LISA(I)
        H(J) ≥ BIG+1
        Y: H(J):=0
        N: F(J):=C(J)
INITIAL
    FOR I:=1 TO N
        RC(I):=0
    K:=0,ST:=0,ITER:=0,ITERD:=0
OPTIMAL
    F(M) > 0
    Y: --->CHECKIN
    N: RETURN
CHECKIN
    FOR I:=1 TO N
        G(I):=0,IK(I):=0
    IC:=0,CK:=0,Z:=0,ME:=99999,NK:=0
    --->LEAVE
```

Subroutine PARARC

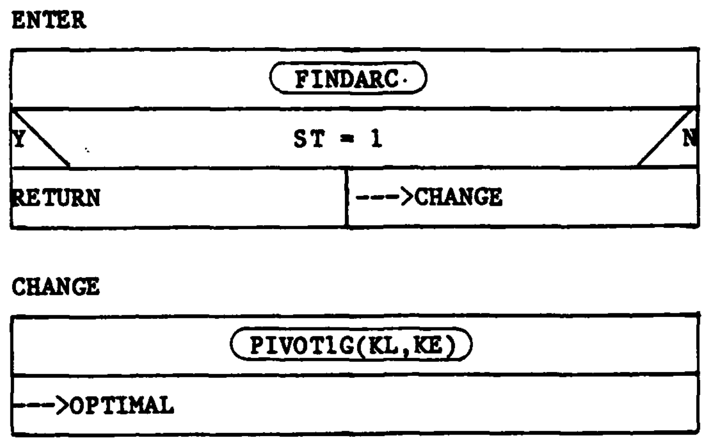
ence number. Store the smallest reference number ME. If the parameter is negative, is set IE to the terminal node of XK, otherwise IE is the origin node of XK. Call PATH which creates LISA,LISN and RC lists for all flow augmenting trails. If a cycle has been found in PATH, call CYCLEG to update γ values for all nodes on the cycle. Call MFLOG2 to determine the maximum reference number and arc which reaches capacity. If KL = 0 any further iterations will drive a capacity below zero. If the reference number is zero, this is a degenerate iteration. Otherwise update flows on basic arcs and flows and capacities on candidate arcs.

```
LEAVE
FOR I:=1 TO XM
//  XKK:=XK(I),ZI:=O(XKK),ZJ:=T(XKK)
//  PB(ZJ) = XKK  OR PB(ZI) = -XKK
//  Y:
//      P(XKK) < 0
//      Y:
//          M1:=F(XKK)/-PARA(I)
//          PX:=-PARA(I)
//          XKE:=-XKK
//          IE:=T(XKK)
//          JE:=O(XKK)
//          M1≥ME
//          Y:
//              ME:=M1,NK:=XKK
//          N:
//      N:
//          XKE:=XKK
//          PX:=PARA(I)
//          IE:=O(XKK)
//          JE:=T(XKK)
//      IJ:=IE,TM:=1,MF:=ME,ITER:=ITER+1
//      (PATH(IJ,TM,XKE,XKK,PX,IK,CK))
//      IJ:=JE,TM:=-1
//      (PATH(IJ,TM,XKE,XKK,PX,IK,CK))
//  N:
Z = 0
Y:
N:
    (CYCLEG)
(MFLOG2)
KL = 0
Y:
    (FLOWG2)
    RETURN
N:
    MF=0
    Y:
        ITERD:=ITERD+1
    N:
        (FLOWG2)
    --->ENTER
```

Subroutine PARARC(cont)

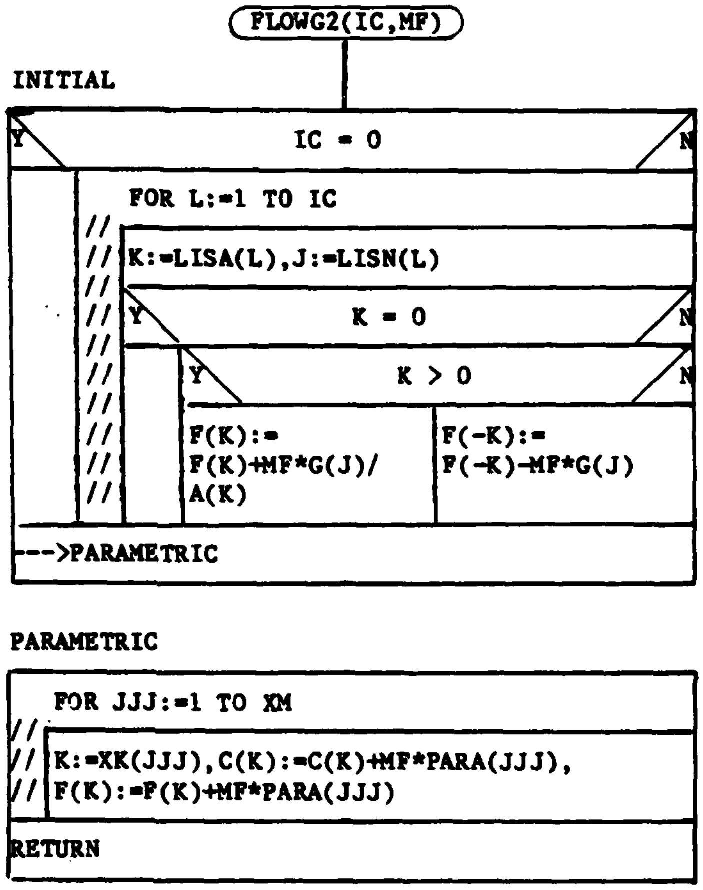
6.(ENTER) Call FINDARC (same as FIND2G except it has been modified for FORTRAN coding) which will determine KE. If KE = 0 there is no feasible solution to the problem and return. Otherwise go to 7.

7.(CHANGE) Change the basis and update dual variables. Go to 3.



Subroutine PARARC(cont)

PURPOSE: To change flow on each basic arc by MF and to update candidate arc flows and capacities.



```
FLOWG2(IC,MF)

INITIAL
IC = 0
Y:
N:
    FOR L:=1 TO IC
        K:=LISA(L),J:=LISN(L)
        K = 0
        Y:
        N:
            K > 0
            Y: F(K):=
               F(K)+MF*G(J)/
               A(K)
            N: F(-K):=
               F(-K)-MF*G(J)
--->PARAMETRIC

PARAMETRIC
FOR JJJ:=1 TO XM
    K:=XK(JJJ),C(K):=C(K)+MF*PARA(JJJ),
    F(K):=F(K)+MF*PARA(JJJ)
RETURN
```

Subroutine FLOWG2

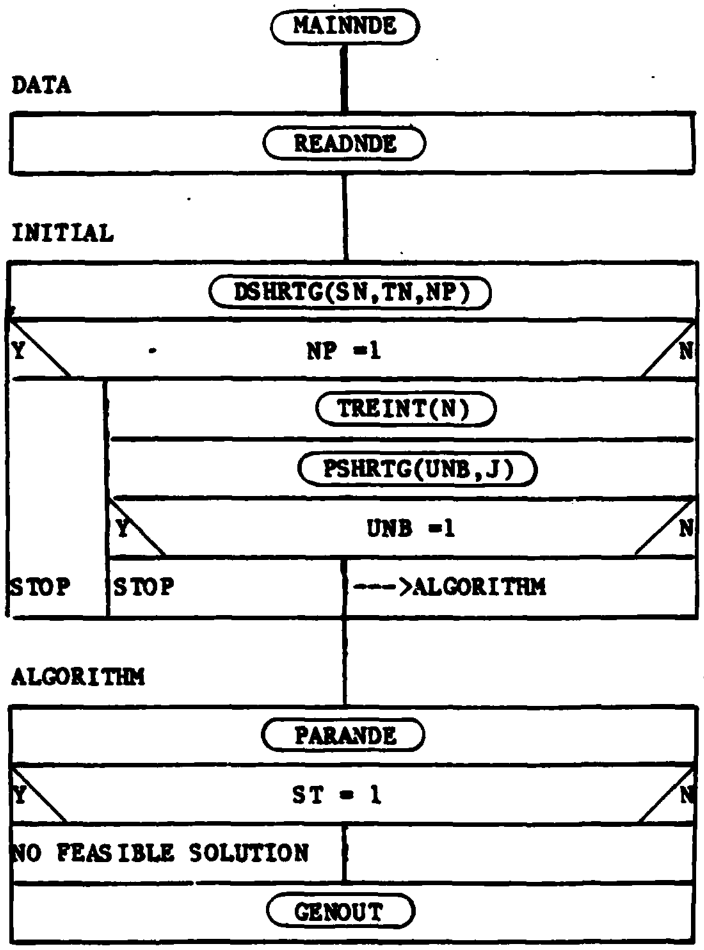
APPENDIX 2

PURPOSE: To read data, initialize the basis, and call parametric node incremental algorithm to generalized network problem.

1.(DATA) Read problem data from input source.

2.(INITIAL) Use DSHRTG to find the shortest path from slack node to all destinations. If no path exists (NP =1), stop. Otherwise use TREINT to construct pointer representation of tree. Call PSHRTG to find shortest path for generalized net work. If UNB =1 an unbounded solution exists. Otherwise go to 3.

3.(ALGORITHM) Call parande to solve problem. If ST = 1 there is no feasible solution. GENOUT prints solution results.



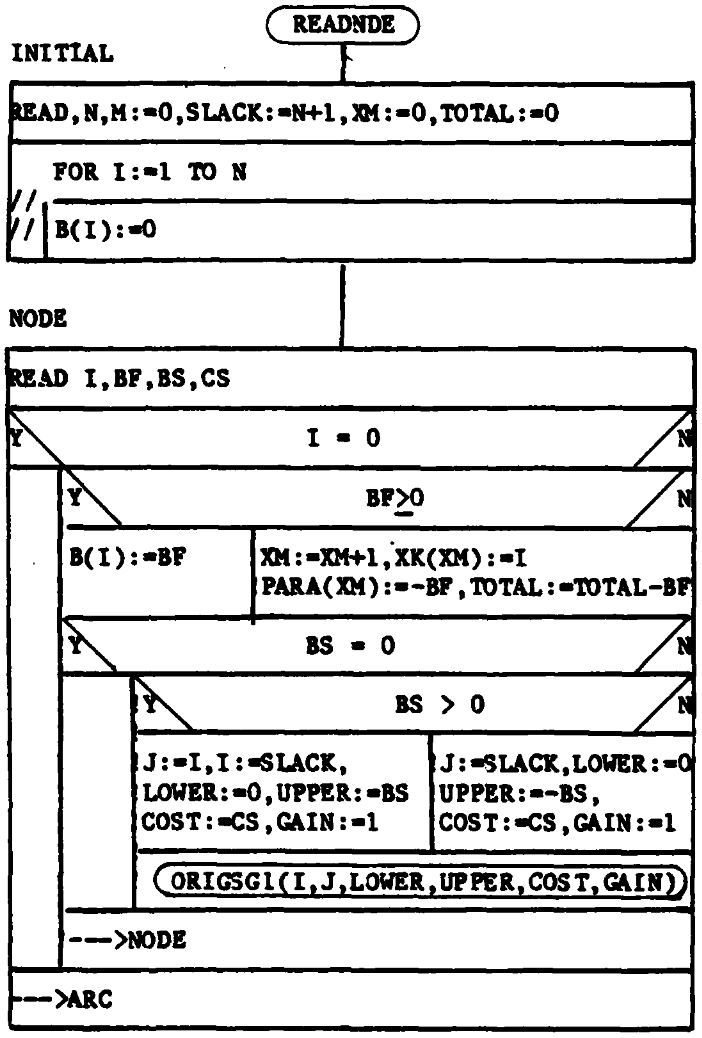
Main Program MAINNDE

PURPOSE: To read and store node and arc data for the minimum cost flow problem.

1.(INITIAL) Initialize the number of arcs to zero, total demand to zero,and number of parmetric nodes to zero. Set SLACK = nodes + 1.

2.(NODE) Read a node data item.If the item is blank go to 3. If the fixed external flow is positive put it in storage . Otherwise set up parametric node list and keep track of total demand.If slack external flow is zero go to 2. Otherwise create slack arc and store data in the correct position in the arc list.

3.(ARC) Read arc data item. If item

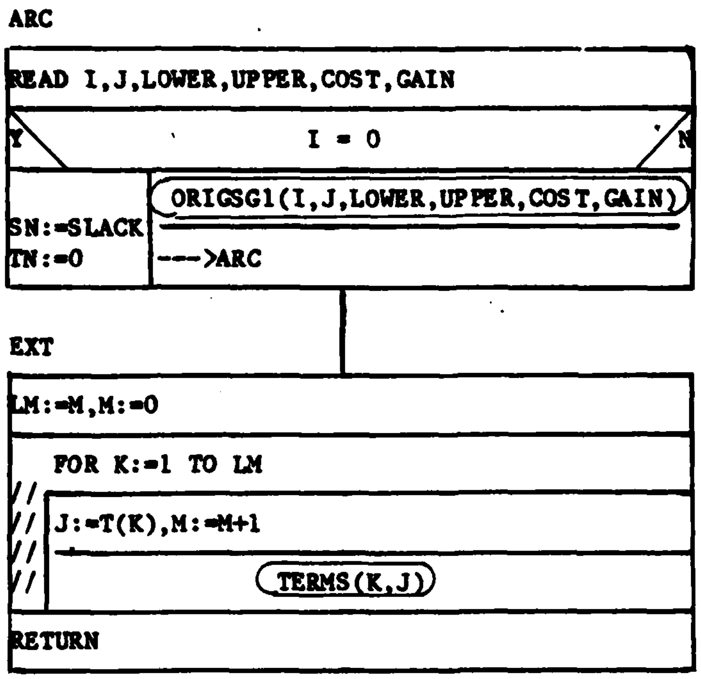


```
READNDE
INITIAL
READ,N,M:=0,SLACK:=N+1,XM:=0,TOTAL:=0
FOR I:=1 TO N
    B(I):=0
NODE
READ I,BF,BS,CS
I = 0
  Y:
    BF≥0
      Y: B(I):=BF
      N: XM:=XM+1,XK(XM):=I
         PARA(XM):=-BF,TOTAL:=TOTAL-BF
    BS = 0
      N:
        BS > 0
          Y: J:=I,I:=SLACK,
             LOWER:=0,UPPER:=BS
             COST:=CS,GAIN:=1
          N: J:=SLACK,LOWER:=0
             UPPER:=-BS,
             COST:=CS,GAIN:=1
        ORIGSG1(I,J,LOWER,UPPER,COST,GAIN)
    --->NODE
  --->ARC
```

Subroutine READNDE

is blank go to 4. Else store the arc in arc list.

4.(EXT) Place each arc's data in the correct position on the arc terminal list.



```
ARC
READ I,J,LOWER,UPPER,COST,GAIN
I = 0
  Y:
    SN:=SLACK
    TN:=0
  N:
    ORIGSG1(I,J,LOWER,UPPER,COST,GAIN)
    --->ARC
EXT
LM:=M,M:=0
FOR K:=1 TO LM
    J:=T(K),M:=M+1
    TERMS(K,J)
RETURN
```

Subroutine READNDE(cont)

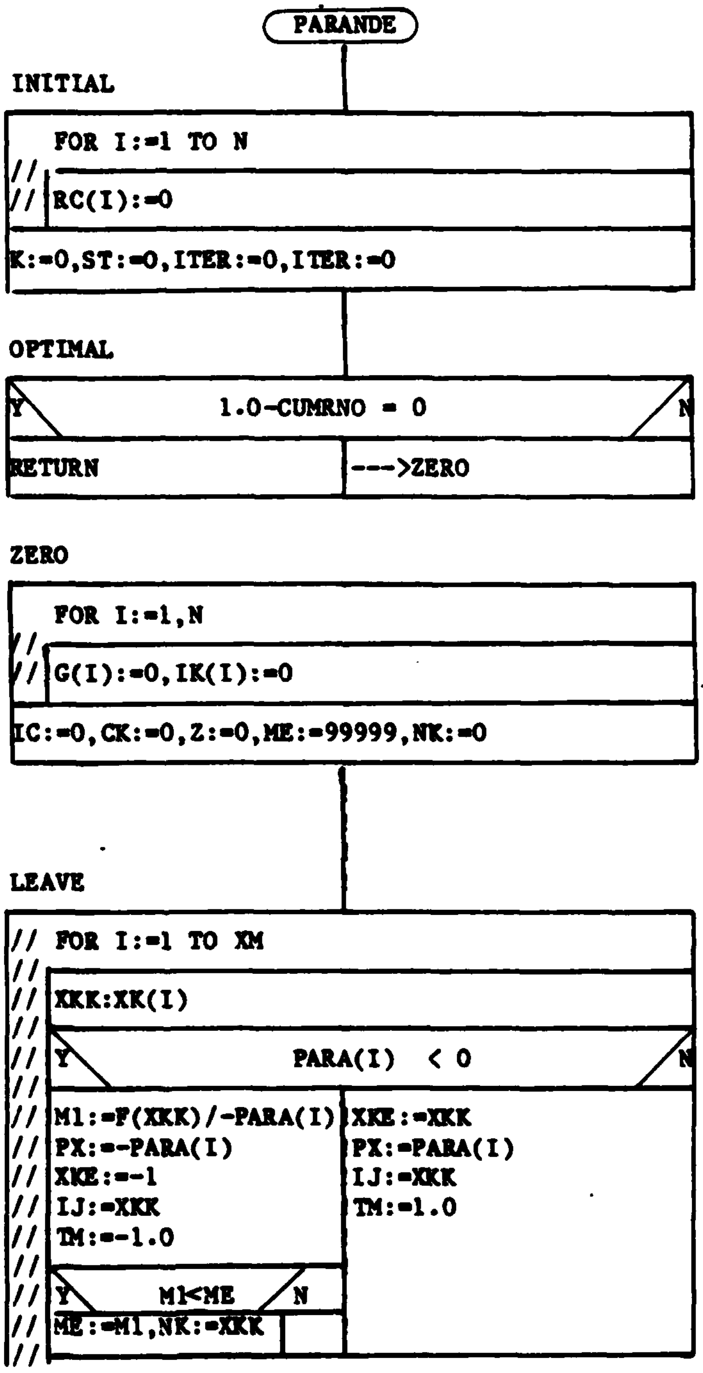
PURPOSE: To implement the parametric node flow augmentation approach for a generalized network.

1.(INITIAL) Initialize all parameters to zero.

2.(OPTIMAL) If the difference between the cumulative reference number and 1 is zero, stop with the optimal solution. Otherwise go to 3.

3.(ZERO) Set the γ's to zero. Initialize arc,node, and cycle counters to zero.

4.(LEAVE) For each node in the candinode list determine if its parametric parameter is less than zero. If so calculate a reference number. Store the smallest as ME. Set IJ to the node number and TM to 1 or -1 as appropriate. Call PATH which will create LISA LISN, and RC lists for all flow augmenting trails. If a cycle(s) has been found in PATH



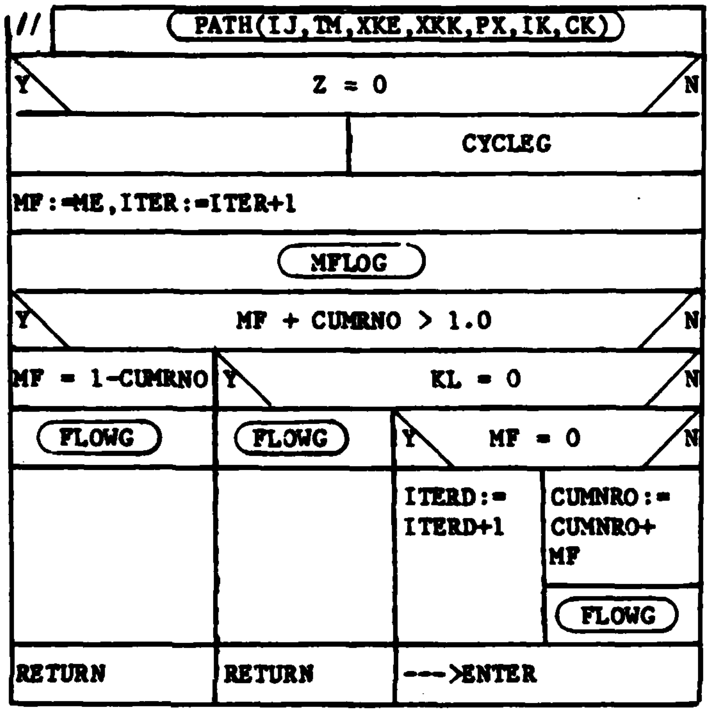
```
PARANDE
INITIAL
    FOR I:=1 TO N
        RC(I):=0
    K:=0,ST:=0,ITER:=0,ITER:=0
OPTIMAL
    1.0-CUMRNO = 0
    Y: RETURN
    N: --->ZERO
ZERO
    FOR I:=1,N
        G(I):=0,IK(I):=0
    IC:=0,CK:=0,Z:=0,ME:=99999,NK:=0
LEAVE
    FOR I:=1 TO XM
        XKK:XK(I)
        PARA(I) < 0
        Y:  M1:=F(XKK)/-PARA(I)
            PX:=-PARA(I)
            XKE:=-1
            IJ:=XKK
            TM:=-1.0
            M1<ME
            Y: ME:=M1,NK:=XKK
        N:  XKE:=XKK
            PX:=PARA(I)
            IJ:=XKK
            TM:=1.0
```

Subroutine PARANDE

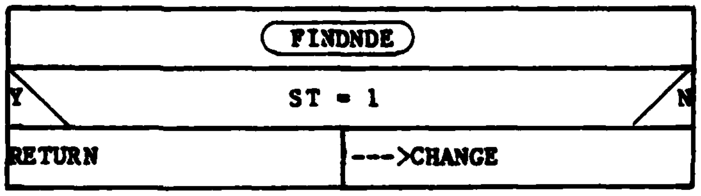
to update γ's for nodes on the cycle(s). Call MFLOG to determine the reference number and the arc which reaches capacity. If MF+CUMRNO > 1 reset MF and update flows and return. If not check if a node will be infeasible at next iteration. If so update flows and return. Otherwise update flows.

4(ENTER) Call FINDNDE which will determine the entering arc. If no arc can be found, there is no feasible solution. Return. Otherwise go to 5.

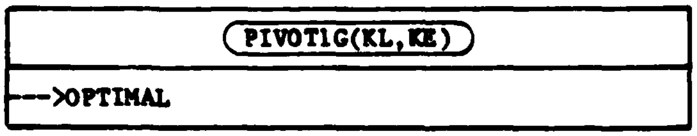
5.(Change) Call PIVOT1G to change basis and up date dual variables.



ENTER



CHANGE



Subroutine PARANDE(cont)

BIBLIOGRAPHY

1. Adolphson, D.L. Private Communication to Michael E. Baum, (January 1984).

2. Adolphson, D.L. "Design and Implementation of a Generalized Network Computer Code," presented at ORSA/TIMS meeting, (November 1980).

3. Ahuja, R.K., et. al. "Parametric Network Feasible Problem," CCERO, Vol. 25 (1983), 13-21.

4. Ali, A., et. al. "Primal Simplex Network Codes: State of the Art Implementation Technology," Networks, Vol. 8 (1978), 315-339.

5. Balachandran, V. and Thompson, G.L. "An Operator Theory of Parametric Programming for the Generalized Transportation Problem: I - Basic Theory," Naval Research Logistics Quarterly, Vol. 22 (1975), 79-100.

6. Balachandran, V. and Thompson, G.L. "An Operator Theory of Parametric Programming for the Generalized Transportation Problem: II - Rim, Cost, and Bound Operators," Naval Research Logistics Quarterly, Vol. 22 (1975), 101-125.

7. Balachandran, V. and Thompson, G.L. "An Operator Theory of Parametric Programming for the Generalized Transportation Problem: III - Weight Operators," Naval Research Logistics Quarterly, Vol. 22 (1975), 297-315.

8. Balachandran, V. and Thompson, G.L. "An Operator Theory of Parametric Programming for the Generalized Transportation Problem: IV - Global Operators," Naval Research Logistics Quarterly, Vol. 22 (1975), 317-339.

9. Balachandran, V., Srinivasan, V. and Thompson, G.L. "Applications of the Operator Theory of Parametric Programming for the Transportation and Generalized Transportation Problems," Mathematical Programming Study, No. 15, 58-85. Amsterdam, Holland: North Holland Publishing Company, 1981.

10. Balachandran V. "Generalized Transportation Networks with Stochastic Demands: An Operator Theoretical Approach," Networks, Vol. 9 (1979), 169-184.

11. Balas, E. and Ivanescu, P.L. "On the Generalized Transportation Problem," Management Science, Vol. 11 (1964), 188-202.

12. Balas, E. "The Dual Method for the Generalized Transportation Problem," Management Science, Vol. 12 (1966), 555-568.

13. Bazarra, M.S. and Jarvis, J.J. Linear Programming and Network Flows, New York: John Wiley and Sons, 1977.

14. Bradley, G.H. "Survey of Deterministic Networks," AIIE Transactions, Vol. 7, No. 3 (September 1975), 222-234.

15. Bradley, G.H., et. al. Applied Mathematical Programming, Massachussetts: Addison-Wesley Publishing Company, 1977.

16. Bixby, R.E. and Cunningham, W.H. "Converting Linear Programming to Network Problems," Mathematics of Operations Research, Vol. 5, No. 3 (August 1980), 321-357.

17. Charnes, A. and Cooper, W.W. Management Models and Industrial Applications of Linear Programming, New York: John Wiley and Sons, Inc., 1961.

18. Charnes, A., Cooper, W.W. and Mellon, B. "Blending Aviation Gasolines: A Study in Programming Interdependent Activities in an Integrated Oil Company," Econometricia, Vol. 20, No. 2 (1952).

19. Charnes, A., Cooper, W.W. and Henderson, A. An Introduction to Linear Programming, New York: John Wiley and Sons, Inc., 1953.

20. Cunningham, W.H. "A Class of Linear Programs Convertible to Network Problems," Operations Research, Vol. 31, No. 2 (March - April 1983), 387-391.

21. Dantzig, G.B. Linear Programming and Extensions, New Jersey: Princeton University Press, 1963.

22. Dantzig, G.B. and Van Slyke, R.M. "Generalized Upper Bounding Techniques," Journal Computer System Science, Vol. 1 (1967), 213-226.

23. Dial, R., Glover, F., Karney, D. and Klingman, D. "A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding the Shortest Path Tree," Networks, Vol. 9 (1979), 215-248.

24. Eisemann, K. "The Generalized Stepping Stone Method for the Machine Loading Model," Management Science, Vol. 11 (1964), 154-176.

25. Elam, J., et. al. "A Strongly Convergent Primal Simplex Algorithm for Generalized Networks," Mathematics of Operations Research, Vol. 4, No. 1 (February 1979), 39-59.

26. Ford, L.K. and Fulkerson, D.K. Flows in Networks, New Jersey: Princeton University Press, 1962.

27. Forrest, J.J.H. and Tomlin, J.A. "Updated Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method," Mathematical Programming, Vol. 2, 263-278. Amsterdam: North Holland Publishing Company, 1972.

28. Fourier, J.B.J. "Solution d'une Question Particular du Clacul des Inequalities Oeuvers II," (1826), 317-328.

29. Gal, T. Postoptimal Analysis, Parametic Programming, and Related Topics, London: McGraw Hill, Inc., 1979.

30. Gass, S.I. and Saaty, T.L. "The Computational Algorithm for the Parametric Objective Function," Naval Research Logistics Quarterly, Vol. 2 (1955), 38-45.

31. Gass, S.I. and Saaty, T.L. "Parametric Objective Function 2," Operations Research, Vol. 3 (1955), 395-401.

32. Gibby, D., et. al. "A Comparison of Pivot Selection Rules for Primsl Simplex Based Network Codes," Operation Research Letters, Vol. 2, No. 5 (December 1983).

33. Golden, B.L. and Magnanti, T.L. "Deterministic Network Optimization: A Bibliography," Networks, Vol. 7 (1977), 149-183.

34. Glover, F, et. al. "Solving Singly Constrained Transhipment Problems," Transportation Science, Vol. 12, No. 4 (November 1978), 277-297.

35. Glover, F. and Klingman, D. "The Simplex Son Algorithm for LP/Embedded Network Problems," Mathematical Programming Study No. 15, 148-176. Amsterdam: North Holland Publishing Company, 1981.

36. Glover, F. and Klingman, D. "A Note on Computational Simplications in Solving Generalized Transportation Problems," Transportation Science, Vol. 7 (1973), 351-361.

37. Glover, F., Klingman, D. and Stutz, J. "Extensions of the Augmented Predecessor Index Method to Generalized Network Problems," Transportation Science, Vol 7. (1973), 377-384.

38. Glover, F., Hultz, J., Klingman, D. and Stutz, J. "Generalized Networks: A Fundamental Computer Based Planning Tool," Management Science, Vol. 24, No. 12 (August 1978).

39. Glover, F., et. al. "Computational Study of an Improved Shortest Path Algorithm," Networks, Vol. 14, No. 1 (1984).

40. Hadley, G. Linear Programming, Massachusetts: Addison-Wesley Publishing Company, Inc., 1962.

41. Hellerman, E. and Rarick, D. "Reinversion with the Preassigned Pivot Procedure," Mathematical Programming, Vol. 1, 195-216. Amsterdam: North Holland Publishing Company, 1971.

42. Hu, T.C. Integer Programming and Network Flows, Massachusetts: Addison-Wesley Publishing Company, Inc., 1970.

43. Hultz, J. and Klingman, D. "An Advanced Dual Basic Feasible Solution for a Class of Capacitated Generalized Networks," Operations Research, Vol. 24, No. 2 (March-April 1976).

44. Jensen, P.A. and Bhaumik, G. "A Flow Augmentation Approach to the Network with Gains Minimal Cost Flow Problem," Management Science, Vol. 23 (1977), 631-643.

45. Jensen, P.A., et. al. "Network Flow Optimization for water Resources Planning with Uncertainties in Supply and Demand," Technical Report CRWR-172, University of Texas (July 1980).

46. Jensen, P.A. and Barnes, J.W. Network Flow Programming, New York: John Wiley and Sons, Inc., 1980.

47. Jewell, W.S. "Optimal Flow through a Network with Gains," Operations Research, Vol. 10 (1962).

48. Johnson, E.L. "Networks and Basic Solutions," Operations Research, Vol. 14 (1966), 619-623.

49. Kantorovich, L.V. "On the Translocational Masses," Compt. Rendu Acad. Sci., USSR, Vol. 37 (1942), 199-201.

50. Kantorovich, L.V. "The Application of Mathematical Methods to Problems of Freight Flow Analysis," Akademia Nauk, USSR, 1949.

51. Kennington, J.L. "A Survey of Linear Cost Multicommodity Network Flows," Operations Research, Vol. 26, No. 2 (March-April 1978), 209-236.

52. Klingman, D. and Russell, R. "Solving Constrained Transportation Problems," Operations Research, Vol. 23, No. 1 (January-February 1975), 91-106.

53. Klingman, D., Napier, A. and Stutz, J.D. "NETGEN - a Program for Generating Large Scale (Un)capacited Assignment, Transportation, and Minimum Cost Flow Network Problems," Management Science, Vol. 20 (1974), 814-822.

54. Koopmans, T.C. Activity Analysis of Production and Allocation, Cowle Commission Monograph Number B, New York: John Wiley and Sons, Inc., 1951.

55. Lasdon, Leon S. Optimization Theory For Large Systems, New York: MacMillian Company, 1970.

56. Lemke, C.E "The Dual Method of Solving the Linear Programming Problem," Naval Research Logistics Quarterly, Vol. 1 (1954), 36-47.

57. Lourie, J.R. "Topology and Computation of the Generalized Transportation Problem," Management Science, Vol. 11 (1964), 177-187.

58. Matsumoto, J. Private Communication to P. A. Jensen, (June 1983).

59. Maurras, J.F. "Optimization of the Flow Through Networks with Gains," Mathematical Programming, Vol. 3 (1972), 135-144.

60. McBride, R.D. and Gupta, A. "Solving Embedded Generalized Network Problems," Paper presented at Fall ORSA/TIMS San Diego Meeting, (October 1982).

61. Minieka, E. "Optimal Flow in a Network with Gains," INFOR, Vol. 10 (1972).

62. Murtagh, B.A. Advanced Linear Programming, New York: McGraw-Hill, Inc., 1981.

63. Neyman, J. and Pearson, E.S. "Contributions to the Theory of Testing Statistical Hypothesis," Statist, Res. Mem., Parts I and II, (1936 and 1938).

64. Orchard-Hay, W. "A Composite Simplex Algorithm - II," The Rand Corporaton Research Memorandum RM-1275, (May 1954).

65. Ramakrishnan, K.G. "Solving Two-Commodity Transportation Problems with Coupling Constraints," Journal of the Association for Computing Machinery, Vol. 27, No. 4 (October 1980), 736-757.

66. Saaty, T.L. and Gass, S.I. "The Parametric Objective Function 1," Operations Research, Vol. 2 (1954), 316-319.

67. Schmidt, S.R., Jensen, P.A. and Barnes, J.W. "An Advanced Dual Incremental Network Algorithm," Networks, Vol. 12 (1982), 475-492.

68. Schrage, L. "Implicit Representation of Variable Upper Bounds in Linear Programming," Mathematical Programming Study No. 4, 118-132. Amsterdam: North Holland Publishing Company, 1975.

69. Schrage, L. "Implicit Representation of Generalized Variable Upper Bounds in Linear Programming," Mathematical Programming, Vol. 14, No. 1 (1978), 11-20.

70. Shapiro, J.F. "A Note on the Primal-Dual and Out-of-Kilter Algorithms for Network Optimization Problems," Networks, Vol. 7 (1977), 81-88.

71. Shapiro, J. Mathematical Programming: Structures and Algorithms, New York: John Wiley and Sons, Inc., 1979.

72. Srinivasan, V. and Thompson, G.L. "An Operator Theory of Parametric Programming for the Transportation Problem - I," Naval Research Logistics Quarterly, Vol. 19 (1972), 205-225.

73. Srinivasan, V. and Thompson, G.L. "An Operator Theory of Parametric Programming for the Transportation Problem - II," Naval Research Logistics Quarterly, Vol. 19 (1972), 227-252.

74. Srinivasan, V. and Thompson, G.L. "Cost Operator Algorithms for the Transportation Problem," Mathematical Programming, Vol. 12 (1977), 372-391.

75. Todd, M.J. "An Implementation of the Simplex Method for Linear Programming Problems with Variable Upper Bounds," Mathematical Programming, Vol. 23, No. 1 (1982), 34-49.

76. Tomlin, J.A. "A Parametric Bounding Method for Finding a Minimum L_∞-Norm Solution to a System of Equations," Technical Report Sol. 75-12, Systems Optimization Laboratory, Stanford University, May 1975.

77. Tomlin, J.A. Private Communication to Michael Baum, (December 1983).

78. Von Neuman, J. and Morgenstein, O. Theory of Games and Economic Balance, Princeton, New Jersey: Princeton University Press, 1944.

VITA

Michael Ernest Baum was born on 20 February, 1951 in Poughkeepsie, New York, the son of Rolf and Tony H. Baum. He graduated salutatorian from Hillsboro-Deering Cooperative High School, Hillsboro, New Hampshire in 1969 and subsequently attended the University of New Hampshire at Durham. In 1973, he graduate Magna Cum Laude with a Bachelor of Science in Mathematics and was commissioned a Second Lieutenant in the United States Air Force. He then attended Case Western Reserve University in Cleveland, Ohio where he received his Master of Science in Operations Research in 1975. Capt Baum entered active duty in May 1975, received his pilot aeronautical rating in 1976 at Reese AFB, Lubbock, Texas, and was then assigned to the 69th Bombardment Squadron at Loring AFB, Maine where he piloted a B-52G heavy strategic bomber. In 1981 he was assigned to The University of Texas at Austin to complete his doctoral degree in Operations Research through the department of Mechanical Engineering. Capt Baum is a member of Phi Beta Kappa, Phi Kappa Phi, and the Operations Research Society of America. In 1983 he married Sigrid Levi.

Permanent Address	1231 Sorrento Road Florence, Alabama 35360

This dissertation was typed by The Baum Corporation.